DOCUMENT RESUME

ED 368 263 HE 027 279

TITLE	Managing Information Technology as a Catalyst of Change. Track IV: Managing in a Client/Server Environment.
INSTITUTION	CAUSE, Boulder, Colo.
PUB DATE	94
NOTE	87p.; In: Managing Information Technology as a Catalyst of Change. Proceedings of the CAUSE Annual Conference (San Diego, CA, December 7-10, 1993); see HE 027 275.
AVAILABLE FROM	CAUSE Exchange Library, 4840 Pearl East Circle, Suite 302E, Boulder, CO 80303 (individual papers available to CAUSE members at cost of reproduction).
PUB TYPE	Speeches/Conference Papers (150) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	Computer Networks; *Computer Oriented Programs; Computer Software; Educational Change; Fund Raising; Higher Education; *Information Management; Information Systems; *Information Technology; Leadership; *Management Information Systems; Program Development; Program Implementation
IDENTIFIERS	*CAUSE National Conference; *Client Server Computing Systems; Colorado State University; Computer Security; Dartmouth College NH; Harvard University MA; Indiana University; University of Chicago IL; University of Massachusetts; University of Saskatchewan (Canada)

ABSTRACT

Eight papers are presented from the 1993 CAUSE Conference's track on effective management of information technology at colleges and universities using a client/server environment. Papers include: (1) "Moving to Client/Server Application Development: Caveat Emptor for Management" (William Barry), which offers an overview of two mature client/server applications developed at Dartmouth College (New Hampshire); (2) "Client/Server as a Software Architecture" (Alan J. Deschner), which describes the application system designed by the University of Saskatchewan; (3) "SOLAR: Harvard's Client/Server Based Fundraising Management System" (James Conway and others); (4) "Talking Turkey about 'Real Change'" (Carole Cotton), which summarizes survey findings concerning changes in computing models; (5) "Desktop Information Delivery for Effective Administration: Client/Server Solutions" (Gary M. Hammon), which discusses applications at the University of Massachusetts; (6) "From Server to Desktop: Capital and Institutional Planning for Client/Server Technology" (Richard Monty Mullig and Keith W. Frey), which examines planning and implementation at the University of Chicago Biological Sciences Division; (7) "Right Sizing a Mainframe Administrative System Using Client/Server" (Ron Dawe and Robert Cermak), which deals with the administrative computing environment at Colorado State University; and (8) "Security in a Client/Server Environment" (Gerry Bernbom and others), which considers the design of information security solutions at Indiana University. (Some papers contain references.) (JDD)

ED 368 263

Managing Information Technology as a Catalyst of Change

Proceedings of the 1993 CAUSE Annual Conference

TRACK IV

MANAGING IN A CLIENT/SERVER ENVIRONMENT

December 7-10
Sheraton on Harbor Island
San Diego, California

HE027279

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

V. Ryland
CAUSE

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it
☑ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

BEST COPY AVAILABLE

TRACK IV

MANAGING IN A CLIENT/SERVER ENVIRONMENT

Coordinator: John Sack

Open systems, interoperable systems, client/server systems … these are not only changing the rules for hardware and software, they are changing the paradigm for management as well. The central question for IT managers on campus is how to minimize risks while optimizing rewards when developing systems for these new models.

Moving To Client/Server Application Development: Caveat Emptor for Management

William F. Barry
Director of Administrative Computing
Dartmouth College
6209 Clement Hall
Hanover NH 03755-3574
(603) 646-3601
william.barry@dartmouth.edu

Client/server systems architecture is evolving to include a set of concepts and tools that range from many exemplary success stories to some intriguing but unresolved problems. Much of what is being called, or sold, as client/server is confounded by a substantial amount of confusion due to: still maturing designs, standards and tools; as well as vendor, consultant and colleague aggrandizement. This has resulted in a level of expectations about the benefits and appropriate use of client/server which is confused by many myths, misconceptions and incomplete information.

This paper presents: a summary of management considerations and recommendations involving moving to client/server application development; and an overview of two mature client/server applications developed and extensively used at Dartmouth College.

prepared for presentation at CAUSE93, San Diego, California

December 8, 1993

Introduction

Caveat Emptor; let the buyer beware! Valuable consumer advice, whether a manager is faced with gold or snake-oil in the client/server marketplace.

The preferred architecture of computing applications is continuing to evolve. Made possible by advances in hardware, software and network technologies, the concepts of "Client/Server" computing represent an emerging technology which offers valuable design features worthy of immediate use for many applications. From a perspective of most mainframe modeled systems, moving to client/server represents a dramatic transition involving technological and organizational change that can often be far more complex and costly than many vendors or pioneers admit.

The client/server model is still evolving as a set of concepts and tools that range from proven success stories to some intriguing but unresolved application software and operating system problems. As with any change of the magnitude that some consider a "paradigm shift", most of what is being called, or sold, as client/server is confounded by a substantial amount of confusion due to: unresolved problems; still maturing designs and tools; as well as vendor, consultant and colleague aggrandizement. These problems are exacerbated due to the inexperience of computing professionals who are too busy for, or in some cases incapable of, successful training in this new model. This has resulted in a level of expectations about the benefits and appropriate use of client/server which is confused by many myths, misconceptions and incomplete information.

Resulting from an effort to sort through these issues in order to reach intelligent decisions on information systems investments, this paper presents: a summary of management considerations and recommendations involving moving to client/server application development; and an overview of two mature client/server applications developed and extensively used at Dartmouth College.

Why all the Client/Server Enthusiasm?

The attractiveness of Client/Server systems concepts may be associated with its compelling arguments which seek to better leverage desktop computing resources, further empower the independence of users, and take advantage of increased capacity and reliability of networks. Furthermore, the intriguing technical challenges of continuing advances towards achieving the goals of open systems in distributed or cooperative processing applications pose exciting, and frequently solvable system challenges. These factors, considered in contrast with the burdens and frustrations of many of our older legacy systems, may explain some of the initial apparent willingness of many computing professionals to so fervently embrace client/server as the long awaited promised land of information technology.

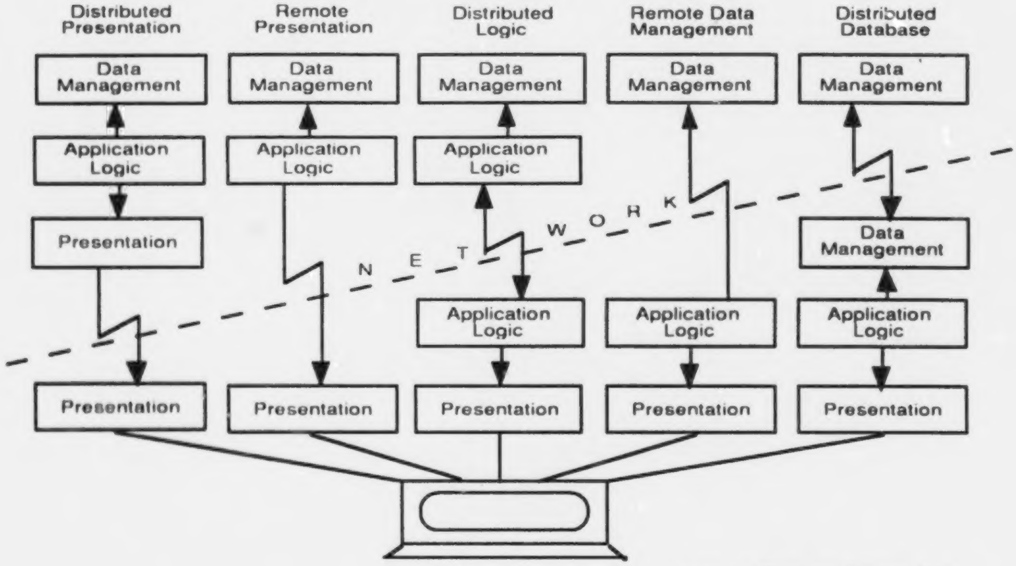
Management decisions involving information technology investments should be driven by prioritized needs, based upon a realistic assessment of costs and benefits. Considerations of the exciting promise of emerging technologies should factor in the wisdom gained from computing's history of exaggerated promises and panaceas. As the initial fervor of client/server enthusiasm has been tempered with experience, our industry has once again been reminded that still maturing technologies need to be approached carefully. Acknowledging that many emerging technologies are built upon compelling and in some cases attractive and thorough systems design principles; the current appropriateness of each of these technologies and trends needs to be judged according to a few criteria. These criteria include: the maturity and reliability of the technology; the costs of adoption; the benefits and tradeoffs relative to other approaches; the availability of enough sufficiently skilled staff; and the balance between a complementary fit with existing systems goals versus the timing of needs to embrace change in existing approaches.

Defining the Client/Server Model

> "What does client/server mean? Is it merely a state of mind, a fashion, a philosophy, an attitude? No; not just. Reports coming back from the bleeding edge tell us that client/server is in fact a *technology;* tough, complex, incomplete, and not inexpensive. After several years of dynamism, the revolutionary fervor that has surrounded client/server is dissipating. Rather than a clear rational construct, client/server is a nest of new and interrelated challenges. As evidenced by its wide popularity... client/server crosses several (in fact, nearly all) technology boundaries: database, applications, networks, systems, and hardware. And perhaps most importantly, client/server implies dramatically new management approaches to gathering, accessing, and maintaining information itself." Stodder (1993)

One of the frequently referenced models of client/server computing was developed by the Gartner Group (1992). This model divides an application into three logical parts (the user interface *presentation*, the business function *processing logic* and the *data management*) and two physical parts (the client system and the server system). The Gartner model outlines five different styles of client/server distinguished by where the network division of the three logical parts of the application occurs.

Styles of Client/Server Computing



source: Gartner Group, Inc.

These five styles of client/server can be summarized as follows

Distributed Presentation: data management, processing logic and presentation components all reside on the server hardware and the presentation component is networked to the users local device (terminal, or desktop computer).

Remote Presentation: data management and processing logic components reside on the server hardware and, separated by the network, the presentation component resides on the users local computer.

Distributed Logic: data management and some processing logic components reside on the server hardware and separated by the network, additional processing logic plus the presentation component resides on the user's local computer.

Remote Data Management: the data management component resides on the server hardware and, separated by the network, the processing logic and the presentation component resides on the users local computer.

Distributed Database: some data management components resides on the server hardware and, separated by the network, additional data management components reside on other server hardware or the users local computer. The processing logic and the presentation components resides on the users local computer .

Depending on a variety of factors (including the application need, the configuration of hardware, network resources and available software tools), any one of these five styles of client server may represent an appropriate and advantageous use of the client server model.

This brief overview of the Gartner Group model is highlighted here only to provide a framework for discussion of this evolving technology. As further client/server experience is gained and the available repertoire of tools evolves, the existing models will be extended or replaced. For example, one variation on the Gartner Group model, reported by Winsberg (1993), states that the issue of distributed database is not directly germane to the model since database management system software should hide the issues of distributed database from the application programmer.

> "Defining client/server seems to be a new kind of parlor game for the industry. I've heard it described as a style of computing, a collection of technologies, an architectural platform, an application development method, a systems integration solution, a re-engineering tool and - heaven help us all - a paradigm shift." Johnson (1993)

Organizational Responsibility and Client/Server

In addition to providing models for various configurations of application components, client/server architecture creates an opportunity to move away from the model of centralized responsibility for developing and maintaining systems. Among its more optimistic and ambitious goals, the vision of client/server facilitating opportunities for the flexible independent creation and support of decentralized computing systems can encourage a new model of decentralized system control and responsibility. These opportunities can compliment and further extend the advantages of decentralized computing staff. However, institutional size and information technology budget levels need to be factored into decisions to move towards decentralized responsibility for development and operations of computing applications. Large institutions which have an appropriate number of decentralized competent systems development staff can be effective in using a client/server application architecture to move towards further distributing a systems total development and operational responsibility. A small university or college, having retained the centralized organizational model of supporting administrative systems, can more economically achieve many of the advantages of client/server systems with no change in who is responsible for systems development and support.

Partial Summary of Anticipated Benefits of Client/Server

- flexibility of independence between application components
- reduced later maintenance costs
- better utilization of lower cost (per MIP) decentralized computers
- elimination of high maintenance costs on older mainframes and minicomputers
- separation of some programming tasks (e.g. presentation) from complexities of network or database management system.
- reduced dependency on one or a few vendors' proprietary systems environments

Summary of Management Considerations and Concerns

The essential issues for management are to understand what information system needs are to be met and what are the costs and success factors involved among possible alternative approaches. An informed management strategy needs to understand: what applications will gain the most from a client/server approach?, what are some of the cost and risk factors?, and when should their organization venture into this approach? Despite the obvious facts just stated, it is surprising how many managers seem willing to buy into a change in systems strategy without attempting to assess the facts.

Cost Issues

Understanding an institution's current level of information technology resources is essential in planning the costs or estimated savings involved in moving to client/server systems. Efforts to plan the costs of client/server applications need to consider: expenses associated with creating or upgrading campus networks; the capacity of installed desktop computers; design complexities of many client/server applications; and the staff learning curve or retraining issues.

Trade press and industry consultants [e.g. Gartner Group (1992); Kennedy, et al (1993); Cafasso(1993); and Anthes (1992)] are increasingly reporting that moving to client/server actually increases costs. As client/server methods and tools mature, and after an organization successfully completes the initial learning curve, many of the costs of creating and supporting these systems will drop and the anticipated long-term benefits are expected to outweigh the costs.

However, Ambrosio (1993) reports that recent studies completed by the Gartner Group present the conclusion that, when considering the total cost of computing, moving from a mainframe-centric model of application development and support to client/server can cost 50% more than a comparable mainframe-based system. It is important to note that this analysis is based upon the current costs of *new* mainframe and mid-range hardware and software licenses, not earlier generations of higher-priced mainframes and astronomical platform-based software pricing. A key cost factor in the Gartner Group's equations is the fact that client/server based systems typically have technician labor support costs, per user or workstation, that are ongoing and are higher than the labor support costs of a centralized system.

But, for some applications, a move to client/server can be quite justifiable and desirable, as a way to better meet the needs of some applications that have compute-intensive or screen I/O intensive needs that are best localized closer to the users desktop. Therefore, moving to client/server should be considered in terms of the improved functional value delivered for some applications, not as an approach to cost savings.

Network Resources

Realizing the potential of Client/Server applications architecture depends upon the completeness of adequate network bandwidth to all desktops to be served. Reliable, high speed, high capacity network services are essential to succeed with client/server. Capabilities of 10Mb/sec network speeds, to the desktop, should be considered a minimum for applications of moderate complexity. If multiple desktop applications will each involve concurrent client/server sessions, then higher bandwidth is needed. To accommodate future applications involving the transmission of digitized voice, video or images, higher network speeds will be necessary (e.g. FDDI at 100 Mb/sec or ATM at 150+Mb/sec).

If a campus is still being served by only asynchronous networks intended to support host to terminal session access requirements, there are many reasons to emphasize network upgrades as a top priority. As a foundation for client/server applications, reliable network services should include a minimum of: network access to all desired campus constituents; comprehensive electronic-mail; print services; network authentication; and file transfer services.

Desktop Resources

In most institutions, the deployment of desktop computers has been incremental over a period of several years of rapid expansion of available desktop computing power. These computers have often been selected according to varying assumptions and understandings of desirable device capacity requirements. This has often resulted in an installed base of not-fully-depreciated desktop equipment, that is insufficient to support the memory, CPU speed, I/O channel or disk requirements of client software and/or data needed as part of a new application. New client/server systems may require upgrades or replacements to some or all desktop PCs. If the desktop device is still a terminal, the costs of acquiring desktop PCs or workstations needs to be factored in.

Other desktop considerations include: the limits of earlier operating systems (e.g. the memory allocation limitation problems of DOS, or the limits on file-sharing of older version of Mac OS). A suggested minimum client CPU is a 386 class Intel chip (or a Macintosh with an 68030-25MHz) and, for low-end server CPUs, at least a 486 class Intel chip (or a 68040).

Standards and Planned Architecture

Successful implementations of client/server applications must be built upon standards of a well-defined systems architecture - if they are to be reliable, scalable, expandable and enduring. A comprehensive strategy defining standard requirements for software, operating systems, data administration, APIs and RPCs, network protocols supported, and hardware configuration is a prerequisite for widespread success with client/server based systems. Such an architecture may include locally defined standards for system component interface requirements or preferably established industry standards. The architecture should include a definition of the intended scope of the problem being addressed and the policies and procedures that will ensure adherence to the architecture's standards.

The very real potentials of interoperability, decentralized independent software development or acquisition, and independence from proprietary systems will fail without success at further establishing and adhering to standards.

Among the shifting sands of vendor consortium or user group standards definition efforts, the Open Software Foundation's Distributed Computing Environment (DCE) standards are emerging as a set of "vendor-neutral" standard APIs (application programming interfaces) that are demonstrating much promise as a cornerstone of software interfaces upon which to implement client/server software. One key remaining weakness of OSF's efforts have been the delays in completing their set of Distributed Management Environment (DME) standards which are intended to address many system and operation management needs. DME is anticipated to be completed in 1995.

Despite the efforts of OSF and many other standards shaping organizations, achieving standards that are comprehensive to any domain, and then accomplishing vendor adherence to those standards is a slow and painful process, at best. Despite the tremendous advances that efforts to achieve open systems have successfully delivered and the examples of successful realization of standards facilitated interoperability, open systems are not yet open! Given the politics of change plus the free-market economic forces upon which a vendor's product differentiation in the marketplace can determine market share and survival, progress towards standards and the commoditization of software components will continue to move ahead slowly, at best.

One way that this can create a problem for client/server is referred to by Roti (1993) as "versionitis... the software malady that causes version 1.2 of product X to work only with version 2.1 of product Y and not with version 2.0 or 2.2. In client/server systems there may be as many as six or seven distinct pieces of software between the user and the database, there may be only one specific version of each of those pieces that works correctly with the others. Change any piece and the whole thing stops working." Considering the lack of sufficient debugging tools in many client/server application development environments, this problem of "versionitis" creates a significant cause for concern. Roti concludes that, given the not-yet-realized promise of 'open systems', it may be wise to minimize the number of vendors involved in components of a client/server application.

Data Considerations

The ideals that many vendors and systems designers are pursuing envision an institution's logically integrated database comprised of networked, distributed application databases created and managed on different but openly accessible software and hardware platforms. According to this model, the integrated databases, some of which may exist on physically separate hardware platforms, can be made to appear to any client program as a single uniform system resource.

Some of the advantages of this model include:

- the ability to accommodate a department's local data needs in ways that coexist with solutions to institutional data needs
- the utilization of less expensive distributed hardware options that can be independently scaled to meet localized processing requirements
- opportunities for reduced network traffic when localized data needs can be met by a departmental server
- increased opportunities to design and tune database and hardware resources towards specialized needs (e.g. high volume transaction updates versus read-only query access)
- greater vendor independence

Some serious disadvantages (or remaining flaws) of this model include:

- the advantages of mixing and matching most current database applications comes at a cost of complexity in making distributed applications work together. In the current state of these technologies, ***transparent interoperability*** is frequently only partially realized.
- the current state of systems management tools to support coordination of distributed heterogeneous databases is both weak and incomplete. Problems of configuration management, operations management, transaction journaling, auditability, contingency planning, security authentication and access controls remain to be properly resolved.
- offsetting the hardware budget savings of less expensive localized departmental servers are the increased labor costs of managing networked distributed hardware.

It is also anticipated that, simultaneous with the maturation of the complex tools and design methods required with distributed databases, there will be a continuation of the trend towards lower cost of larger electronic, magnetic and optical data storage devices, faster and higher capacity data I/O buses and more powerful single and parallel CPUs. More progress towards the outcome of these sets of evolving technologies should be achieved prior to any substantial move towards distributed databases, unless there are other factors to justify such a change.

Issues of data administration should also be considered. As database management system (DBMS) tools continue to mature, much progress has been made in preserving the integrity and consistency of distributed databases. However, prior to worrying about which DBMS vendor has solved "two-phase commit" problems of updating a transaction across a distributed dataset, or before assessing whether a vendor's product can handle data rollback across heterogeneous databases, many organizations should continue efforts to reconcile their lack of successful data administration involving their central systems. These challenges won't get any simpler to solve if databases become more distributed. This is not an argument against client/server. It is a reminder of the need to judge an organizations position relative to newer technology against the degree of success to which more mature methods and technologies have been applied!

Is a reluctance to move to distributed databases running against a real or perceived trend towards "downsizing and decentralizing" all of computing? Yes. Is this a sound position or is this just a resistance to change and a perpetration of the old model by a shortsighted "mainframe bigot"? According to Gillan (1993), when asked about the future of centralized data storage, Bill Gates (founder and CEO of Microsoft) stated, "We're in the information age and that means there'll be a lot of information. There are still large economies of scale in storage costs and administration of centralizing that data, and as fiber brings communication costs down, you'll be able to pool a lot of that data in one place... The productivity application world and the data center world are not separate any more".

Systems Management in a Heterogeneous Environment

In contrast with available systems management tools for use on mini or mainframe computers, an area of concern in the client/server arena involves the lack of complete and reliable tools for a wide array of important systems management tasks. These missing or incomplete tools include operating system or application utilities to support operations scheduling and control; audit tools; rollback journaling; backup and recovery tools; performance monitoring and capacity planning tools; and change control utilities.

Staffing Issues

Experience and skill levels of current IS staff and the costs of new training represent a substantial current obstacle to the adoption of client/server methods. Anthes (1992) reports that the Cambridge Massachusetts research firm Forrester Research Inc. found that 75% of Fortune 1000 firms included in their survey lack the skills needed to work with client/server based systems. Exacerbating this problem is the burden on current staff to support required production maintenance and the ongoing stream of enhancements to existing systems.

While it is wise to encourage the highest potential of our staff, it is also essential to acknowledge limits. Many of the very skills that made some 3GL programmers successful (e.g. linear procedural thinking), seem to get in the way when more abstract reasoning requirements of the multi-layers of software and data manipulation control become involved. Most data processing professionals should be given every chance to make the leap to client/server, and management needs to find ways to fund and nurture that effort. However, there remains the pragmatic reality of some staff who can't or don't choose to keep up with new technologies. For example, consider the difficulties of some staff to move from record oriented to set processing, or the slowness of introducing many software engineering methods, and the difficulty some have with either full or partial data normalization. These past perfomances should give a manager second thoughts about the future career paths of some programmers and systems staff.

Separate from issues of skills retraining, the fact remains that there is a shortage of computing professionals with the proper mix of experience with complex applications development and many of the talents needed to design and build client/server systems.

Some Systems Are Not Appropriate to Current Client/Server Technologies

All forms of centralized systems are not going away. The trend of greater MIPS per dollar spent on smaller machines does not, by itself, dictate that the model of the central computer is no longer of value. In nearly all cases cited in business management or computing trade journals, a business cost analysis which compares the relative cost and capability of mainframe or mini computers to the alternatives are comparing old machines to newer more powerful ones. While the old model of centralized million dollar hardware generating unacceptable recurring maintenance costs to support high-priced application software is being replaced by the competitiveness of newer economies of hardware and software, the appropriate role of centralized computing applications and operations continues to provide economic and operational advantages for many institutional administrative information systems needs. The lower prices of desktop and workstation software are forcing down the previously astronomical pricing structures associated with mainframe or minicomputer software. The marketplace is forcing most applications vendors to shift to per-user software licensing rather than purely machine-sized pricing. The systems management tools necessary to administer distributed systems do not yet compare favorably with those available on single clustered systems.

For small universities and colleges, having most business functions located within a single campus, centralized systems continue to provide the most efficient economies of scale for many institutional administrative systems.* When costs of decentralized hardware, dis-economies of scale in vendor negotiation, programming or support staff, and non-coordinated data definition are considered, many distributed systems are proving to be more costly than centrally developed and managed systems.

As Blythe (1993) reported in the CAUSEASM listserv's summary of an electronic roundtable discussion on "Client-Server Computing: Management Issues"

> Ted Klein, President of the Boston Systems Group writes
> that there are five types of systems that are NOT applicable
> to downsizing with today's technologies:
>
> 1. applications with large databases which cannot be easily partitioned and distributed
> 2. applications that must provide very fast database response to thousands of users
> 3. applications that are closely connected to other mainframe applications
> 4. applications that require strong, centrally managed security and other services
> 5. applications that require around-the-clock availability

Software Development Considerations

A software development team can move towards client/server by establishing and beginning to encourage adherence to a set of client/server modeled software, data access and user presentation design recommendations. These recommended design features, refined by experiences with pilot applications, can begin to incorporate OSF model concepts for RPCs, APIs, and other utilities of the DCE model. In addition, good modular design of software (for any application architecture) should incorporate divisions of programming code between: GUI or screen presentation; SQL statements; business rules; and the linkages between the screen I/O and business rules. Efforts should also be made to experiment with PC, Macintosh or workstation 'front-end' tools that use SQL to access host system databases.

Consideration should be given to learning more about client/server success stories and the opportunities to partner, for example, with the efforts of Dartmouth College's DCIS or Cornell's Mandarin project.

In considering new software tools:

- Attempt to minimize the varieties of DBMS and GUI vendors until there is further progress on standards efforts.
- Favor vendors or programming tools which support industry efforts towards standard protocols (e.g. SQL, OSF's DCE RDA s and APIs). Assess vendors' demonstrable commitment to standards and gateways with proprietary DBMS products.
- DBMS system recoverability: look for working solutions to two-phase commits needs, or data rollback across heterogeneous databases
- Query optimization: be cautious about the proposed database designs creating the distribution of databases that may need to be joined for query. The technologies of query optimization, still maturing within most single vendor DBMS tools, have yet to adequately address query optimization across multiple vendor DBMS.

---

* It should be noted that the model of centralized systems can be consistent with goals of increasing departmental access to and control of data and systems resources.

Overview of Two Successful Dartmouth College Client/Server Projects

Dartmouth College's BlitzMail® electronic mail system.
Placed into production in June 1988, BlitzMail was created to provide a GUI electronic mail system for a primarily Macintosh equipped user community. Currently serving 17,000 members or affiliates of the Dartmouth Community, this client/server based system is capable of 1000 simultaneous client connections and recently hit a new usage peak of 73,000 messages sent in one day.

Connecting more than 7000 Macintosh-based clients, spanning over 200 AppleTalk LANs and a TCP/IP internet link connecting UNIX and XWindows clients as well, BlitzMail is a ubiquitous part of student, staff and faculty work at Dartmouth College. The GUI client software was written in Pascal, the server software was written in C. The server hardware currently employs five Next machines; we plan to install a DEC Alpha workstation as a sixth server in late December 1993. This single new server should provide the capacity to handle an additional 500 simultaneous users. Periodically, upgraded versions of the Macintosh client are distributed using BlitzMail itself. Such upgrades are also accessible for users to copy to their desktop Macintosh computer from a public AppleShare file server.

In addition to a full complement of electronic mail features, BlitzMail allows arbitrary Macintosh documents (such as word processing or spreadsheet documents) to be sent along with mail messages as enclosures. BlitzMail also provides a Macintosh interface to bulletin boards containing "semi-official" information about groups and departments around the campus. BlitzMail acts as a client to a standard NNTP server which is part of our normal Usenet news system.

Dartmouth College Information System (DCIS).
Dartmouth College, with the support of Apple Computer, Inc., has developed an integrated campus-wide information system. DCIS is a client/server architected system which provides an average of 700 users per day with access to over 60 local databases and hundreds of off-campus Internet database resources. DCIS provides access to data from a variety of sources, including: reference encyclopedias; indexes to the Dartmouth College Library's collections and the journal literature; scholarly resources such as the Oxford English Dictionary, a library of commentaries on Dante's Divine Comedy; administrative systems data resources such as the central supplies inventory; and resources like Books in Print.

The software architecture uses the OSI model of communications protocol stacks, incorporating the Z39.50 search & retrieval standard, WAIS protocols, and other locally developed protocols. DCIS databases are distributed across several mainframes and servers spanning multiple operating systems and database management systems. Written in C++, the viewer (client) portion of DCIS, is distributed to thousands of users. DCIS has a self update capability and is also distributed via Dartmouth's blitzmail and from a AppleShare public file server.

The DCIS system and tool set are available to be exported to other environments, including other academic institutions. The available products include: viewer application software; search and retrieval protocol software (a WAIS gateway, a Z39.50 gateway, and a Telnet connection server); servers for BRS, PAT and SPSS database managers; and an authentication server (IDAP).

Conclusions

The following conclusions can be reached about current client/server methods and technologies:

- client/server is real, desirable and inevitable for many applications
- it should be seen as a means to better meet user needs, not as a way to cut costs
- for most applications it is neither simple, nor cheap
- central administrative computing departments should provide leadership in realizing its benefits and optimizing an organizations chances of success with client/server
- it is not a panacea and is not, for the foreseeable future, appropriate for all applications
- it is a fairly immature, emerging technology full of risks and unresolved pitfalls.

References & Bibliography

Ambrosio, Johanna. "Client/server costs more than expected", Computerworld, October 18, 1993, p. 28.

Anthes, Gary. "Middleware Eases App Development", Computerworld, December 29, 1992, p. 113

Anthes, Gary. "Training Biggest Obstacle in Client/server Move", Survey Says, Computerworld, December 14, 1992, p.89

Atre, Shaku and Storer, Peter M. "Client/server Tell-all: 10 things you should know before you tackle your downsizing project", Computerworld, January 18, 1993

Bond, Elaine. "Danger! There's Nobody to Steer Client/server", Computerworld, April 26, 1993, p. 33

Brentrup, Robert J. "Building a Campus Information Culture", CAUSE92 Conference Proceedings, December, 1992

Cafasso, Rosemary. "Client/Server Strategies Pervasive", Computerworld, January 25, 1993, p. 47

CauseASM listserv. Mediated Discussion of the Management Issues of Client/Server Computing, a report summarizing the June through October 1992 electronic listserv discussion involving computing representatives of 41 colleges or universities.

Computerworld. Computerworld "Guide to Client/Server", April 26, 1993, pp. 73-84

Cornell Information Technologies. "Client/Server Applications at Cornell, Doing it Fast and Right", 1992

Finkelstein, Richard. "RDBMS for Client/server: They Don't Quite Measure Up", Computerworld, February 8, 1993, p. 59

Gantz, John. "Is Client/Server a lot of Hot Air?", Computerworld, December 21, 1992, p. 25

Gartner Group. "Strategic Planning Assumptions for Information Technology Management", summary of Cause92 conference presentation, December, 1992

Gillin, Paul. "Not Dead Yet", Computerworld, May 10, 1993, p. 30

Johnson, Maryfran. Computerworld Client/Server Journal, Nov. 1993, vol. 1, number 1, page 3.

Kelly, David. "Training for the Client/Server Leap", Computerworld, May 3, 1993, p.129

Kennedy, Michael and Gurbaxani, Andrew. "When Client/Server Isn't Right", Computerworld, April 19, 1993, p. 99

Keuffel, Warren. "DCE: The Nations of Computing?", Database Programming & Design, Sept. 1993, p. 31-39

Kiernan,Casey. "Client/Server: Learning From History", Database Programming & Design, September 1993, pp. 46-53

Parker, Mike. "A (Relatively) Painless Way to Move to Client/server", Computerworld, June 28, 1993, p. 33

Roti, Steve. "Pitfalls of Client/Server", DBMS , August 1993, p. 67

Stodder, David. "Database: the Next Buzzword?", Database Programming & Design, July 1993, p. 7

Tanenbaum, Andrew S. Modern Operating Systems, (Section 10.2 The Client-Server Model)Prentis Hall, 1992, pp. 402-417

Tobin, Ed. 'Client/server isn't the answer for everything", Computerworld, May 10, 1993, p. 31

Vizard, Michael. "Client/Server Caveats", Computerworld, February 15, 1993, p. 16

Wexler, Joanie. "Stanford Navigates Client/server Peaks, Valleys", Computerworld, June 29, 1992, p. 65

Winsberg, Paul. "Designing for Client/Server", Database Programming & Design, July 1993, p. 29

Client/Server as a Software Architecture

Alan J Deschner
University of Saskatchewan
Saskatoon
Saskatchewan CANADA

Abstract

Adding telephone registration to an already overloaded multi-use computer forced the University of Saskatchewan to consider a client/server architecture. A survey of client/server implementations left us very confused: we encountered everything from PC's accessing database servers, to X-windows systems with the server on the desktop. The only common theme was hardware components on a network.

We came to understand client/server as an interaction between software components. Four distinct components are recognised in a typical operational system: presentation servers, client applications, business servers, and database servers. Each interaction between components has a distinct client and a distinct server side, with the client side being in control. There are general principles governing the function of each component. We call this the 4 box architecture.

In applying this architecture in our project, several trade-offs were necessary to function within our available technology, but the end result was an application system with well defined components that can be re-packaged for different environments in the future.

The Confusion over Client/Server

In late 1991, the University of Saskatchewan embarked on a project to implement a telephone registration system. We started with an Rdb database, an existing over-the-counter registration application written in a 4-GL (DEC Rally), and a new piece of hardware, a Periphonics voice response system. It became clear quite quickly that a straight interface between the new hardware and the 4-GL would overload the host hardware, mostly due to the doubling of the number of consumptive processes running the 4-GL. We turned to client/server ideas as a possible solution.

We discovered that the term *client/server* is very hard to define. To some, it is PC's accessing a database server on a network. To others, it involves remote procedure calls (RPC's) between very sophisticated software components on different network nodes. To vendors, it is anything they can use to sell more product! Most often, it is associated with a particular hardware configuration on a network. In the PC arena especially, it is very hardware oriented: the *server* is usually a database on a dedicated box, and the *clients* run on desktop PC's and contain all the business logic.

The client/server concepts surrounding X-terminals further confused the issue. Here, the X-server is on the desktop delivering display services to the end user, and the application program running on a bigger machine is the X-client! This seemed exactly backwards to the PC situation, where the user's desktop device is the client.

The only feature shared by all these examples is that there is a network connecting two or more machines that are co-operating to accomplish a task. The thing that differentiates one configuration from another is how the work is split among the various processors. At one extreme, we have an IBM mainframe running a CICS *client application*, and block-mode terminals on a 3270-controller (*terminal server*) that handles key-press level events. (I have also seen this described in a way where the terminal is the client and the *big iron* is the server.) The other extreme is a file server on a LAN with the whole application, including DBMS, running on a desktop PC or Mac.

But we were left with the question of what distinguishes *client/server* from other similar concepts, like distributed processing and distributed databases?

A Common Theme

When we examine all these configurations for a common theme, we start to see client/server as an interaction between components, especially software components, more than a hardware configuration. Each pair of software components has a well-defined protocol for interacting with each other. The thing that distinguishes client/server from other architectures is that one side of the interaction is labelled the *client* and the other is the *server* (Figure 1). The question then becomes, what properties do the components on each side have that make them one or the other?

The guidelines we arrived at are:

- The client is in control.
- The server defines the communication protocol and message formats.

- The server does the bidding of the client.
- The server either succeeds or fails in providing the requested service, and leaves things in a well-defined state.
- If the server cannot provide the service, it notifies the client as to why and the client must decide what to do next.

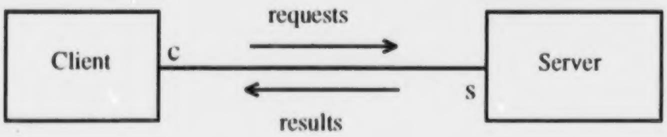

Client
c
requests
results
s
Server

Figure 1 : The Client/Server Relationship

Note that a particular software component can be a server for one component, but a client to a another. Also, not every set of communicating processes meet these criteria; for example, co-routines co-operate to perform a task, but do not have this client/server relationship.

The 4-box Architecture

For operational systems, we define four software components, the "4 boxes", that interact using this definition of a client/server relationship (Figure 2).

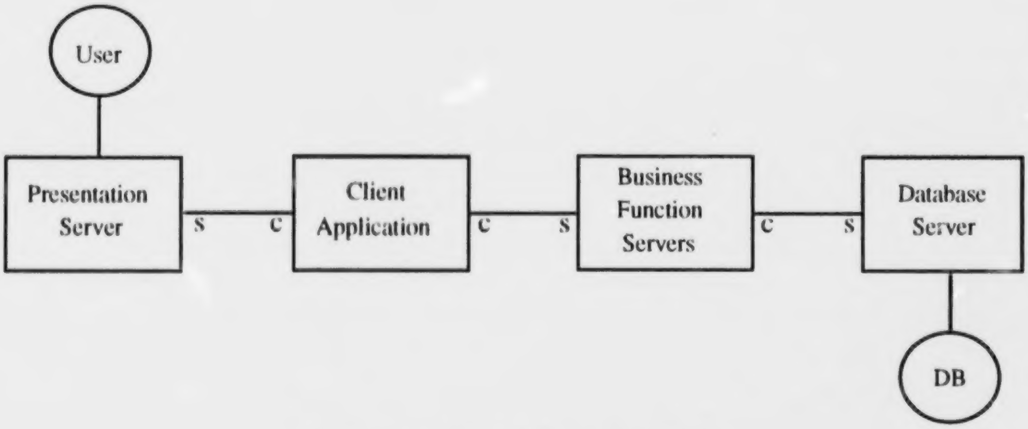

User
Presentation Server
s
c
Client Application
c
s
Business Function Servers
c
s
Database Server
DB

Figure 2 : The 4-Box Architecture

- The **Presentation Server** handles the interaction with the end user of the application: it knows how to communicate with the user, but not the meaning of the communication.
- The **Client Application** is the component in control. It knows both the business functions and the user interface, and makes the business functions available to the user in a controlled way via the interface.
- The **Business Function Servers** provide the basic business functions, and should be sharable among several different client applications.
- The **Database Server** is whatever interface is required to the DBMS, or possibly the DBMS itself.

We generally do not go any lower than the database server, since that is usually where the application programming stops. In reality there could be a file server involved, but it is generally hidden by the DBMS itself.

The Four Components

The presentation server is the component closest to the user, and handles the details of the user interface. It provides its service to the client application. Important examples are X-servers, a "voice server" in an Interactive Voice Response System, a forms handler such as Digital's DECforms, or even a 3270 terminal controller.

The database server is often a relational DBMS such as Rdb, Oracle, DB2, etc. It could, however, be any set of routines that provides access to data at some level above file access. Services are requested by the business servers, often in the form of SQL statements.

A business function server, or simply business server, is a software component that knows the details of the specific business function being delivered. An example of a business function in this context is adding a class for a student, or calculating fees for a student. The main role of the business server is to maintain the integrity of the business data and enforce business rules, for example, ensuring that enrolment and credit unit counters are updated when a class is added, or not letting a student take a class that has been cancelled. Business services are provided to the client application. In turn, business servers are clients of the database server, starting and stopping database transactions, and usually issuing multiple SQL statements to complete their function. In our environment, business servers are stateless, handling each request in isolation from others and never holding database resources between requests, but this need not be the case in general. In many ways, these business servers are like the "transactions" in a transaction processing system (e.g., CICS, ACMS), except that they concentrate on the business issues and ignore the user interface. A business server should be able to deliver its service to multiple clients of different types.

The client application is what ties it all together, and is in control of everything. It uses the presentation server to communicate with the user. It uses various business function servers to do the real work. Where the servers know the specific details of the interactions with the user or the database, the client must know the general nature of both, for example, knowing that the user is on a PC, terminal, telephone, etc. The main role of the client is to make the various business services available to the end user in a controlled way, using the presentation server for communication. It interprets the business server requirements and results to the end user, and interprets the user's requests for the business servers. In our environment, a client application cannot directly access the database, but must use the business servers to get all required information.

Note that what we have called the *client application* is the only "pure" client in the 4-box architecture: it is the only one that is on the client side of its interaction with all other components. Most *servers* are also clients of other components, even if those components do not appear explicitly in our architecture.

The Components as View Translators

There are several different views of what goes on in a computer system, and we can think of each of the four boxes as translating between two such views.

- The client application translates between the user's view and that of the business.
- The business server translates between the business's view and that of the database.
- The database server translates between the database's view and that of the underlying file system or disk controller hardware.
- The presentation server translates between the user's view and that of the underlying presentation/display hardware.

Implementation Issues

In summary, the client/server paradigm can be characterised as having software components that are defined so that each interaction has a well-defined client side and a well-defined server side. The 4-box architecture is one example of how a client/server system can be configured. When we then ask about hardware configurations, we see that this is an architecture that can be distributed across a network in many different ways. In fact, the network can be introduced at any of the interfaces between the four boxes, and where it is placed will generally determine which piece of hardware is called a *client* and which is called a *server*.

An application that requires many SQL statements to process one business function might be best split at the link between the client application and the business servers, so that the business servers are on the same node as the database server. Another application might best be configured with just the database server (network DBMS) on the "server" machine and the other three components on a desktop PC. If the desktop device is an X-terminal, the division is between the terminal server and the client application.

Note that the hardest place to put the network is between the client application and the business server. Communication between a business server and the database server is probably done using SQL, and can be accomplished across the network with tools from the DBMS supplier. Communication between the client application and the presentation server is probably predefined by the nature of the presentation server, for example, if it is an X-terminal. But network communication between the client application and the business servers requires mature middleware that is now only starting to become available in the market-place. Remote Procedure Calls (RPC's) and the Distributed Computing Environment (DCE) are examples of such middleware.

It is, of course, possible for the four components to reside on more than two network nodes. For example, the database server might be on a mainframe, the business servers on a departmental server machine, and the client application on a desktop workstation. Or they might all run on different nodes.

Also, there may be several business servers in a given configuration, and they need not all be on the same node. For example, the business server that registers students might be on a machine in the Registrar's Office while the one that checks for overdue accounts might be in the Business Office, even though both receive requests for service from the same client application. And in these days of multiple and distributed databases, it is likely that a business server itself might have to use database servers on multiple nodes.

Operational Systems vs. Decision Support Systems

An operational system is characterised by very well-defined and often repeated functions that must be performed quickly, such as adding a class. In contrast, a decision support system (DSS) is required to provide very complex processing that is repeated only a few times, such as asking "how much time does the Psychology Department spend teaching Medicine students?" This tends to be ad-hoc read-only processing that has few if any pre-defined access paths, and does not require an immediate response.

The 4-box architecture is not very appropriate for decision support systems. Indeed, it was developed expressly for operational systems, and Telephone Registration in particular. Direct end-user access to the database would seem to be the best way to provide the ad-hoc access required in a DSS. This access is properly provided by a "client application" such as a spreadsheet or a report writer running on a desktop PC and using SQL to access departmental data stored on a network server and/or institutional data on a mainframe. There are no business function servers, and our architecture collapses into just three boxes. Note that this is much closer to what the PC industry generally means by client/server.

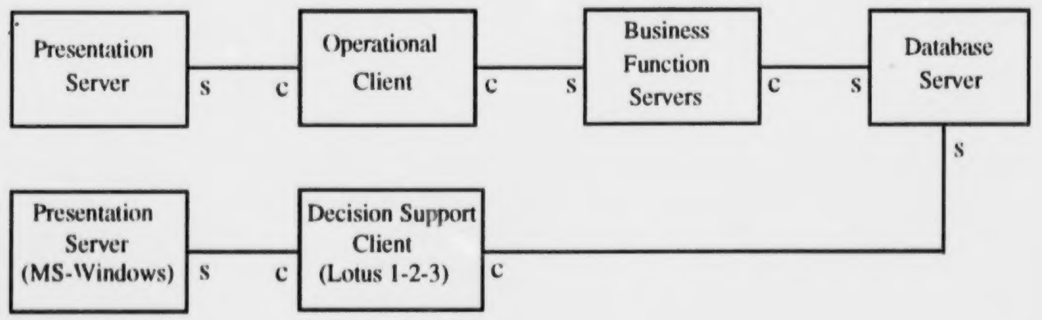

Presentation Server
s
c
Operational Client
c
s
Business Function Servers
c
s
Database Server
s
Presentation Server (MS-Windows)
s
c
Decision Support Client (Lotus 1-2-3)
c

Figure 3 : Operational and Decision Support Clients Co-exist

One direction in the industry is to satisfy DSS needs by building an *information warehouse* as an extract or summary of operational data, and to store it in a relational database on a network server. This is especially attractive when the operational data is stored in a so-called *legacy system* and not available via modern SQL-based tools. However, if the operational system is already based on a relational DBMS and uses the 4-box architecture with stateless servers, there is no reason why it cannot co-exist with DSS's (Figure 3). The only issue might be whether the DSS degrades the performance of the operational system, but that is probably a symptom of a more serious capacity problem with the network and/or the *server* nodes.

Client/Server Transaction Processing

Operational systems are really "lots of the same thing, over and over again", and this is exactly the situation addressed by on-line transaction processing (OLTP) monitors. A natural question is how the 4-box architecture relates to OLTP's.

Basically, we have treated this question as an implementation issue. Some OLTP's like IBM's CICS seem less in line with our architecture, while others like Digital's ACMS seem more so. ACMS has an architecture that on the surface is very much like our four boxes (Figure 4), where the Execution Controller plays the role of our *client application*, and Digital is less explicit about the *database server*. However, it does not permit separation of the client application and the business servers across a network, and we found it to be not completely able to support the level of complexity required in our client application.

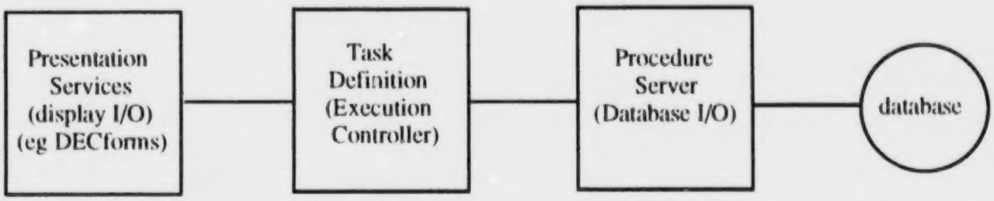

Presentation Services (display I/O) (eg DECforms)
Task Definition (Execution Controller)
Procedure Server (Database I/O)
database

Figure 4 : The Architecture of Digital's ACMS

Our Example: Telephone Registration

The 4-box architecture was developed for use in a Telephone Registration application (TelReg). Our administrative computing environment consists of a VAXcluster with two VAX 6430's and 14 VAXstations running OpenVMS, all connected to the general campus network. We had existing Rdb databases supporting various Rally (4-GL) applications in the Registrar's Office, including on-line registration. An Interactive Voice Response Unit (IVRU) was purchased from Periphonics, with connections to 32 phone lines, a UNIX-like operating system, and application programming in COBOL.

Once the 4-box architecture was developed, we began to look at the implementation issues with respect to TelReg. Specifically, we looked at what middleware could be used to facilitate communications between the various components. We examined Digital's Application Control Architecture (ACA) Services (recently renamed ObjectBroker) which showed great promise in routing messages between clients and servers, and which would allow us to distribute the components in any way we wanted. However, at that time (April, 1992) the product was still immature: we could not find a production installation with an application similar to ours, and we were concerned about our ability to use it effectively.

The decision was made to write our application using ACMS, Digital's transaction processing system described above. The idea was to implement the presentation server on the IVRU in the form of a voice server, which would process *voice forms* sent from the host computer across the network. These voice forms are the voice equivalent of screen forms, with prompts being spoken and input fields entered via the telephone buttons. The client application was to be in ACMS, and the business servers were to be ACMS procedure servers, coded in COBOL/SQL.

We knew this would not allow us to separate the client from the business servers, at least not in the near term. But we still had the possibility of moving the bundled client application and business servers to their own machine, which would then access both the database (via SQL Services) and the voice server across the network. And the ease with witch ACMS handled our inter-process communication needs made it the best alternative.

High level design was done using these ideas. But both ACMS and IVR technology were new to our shop. As we progressed through more detailed design and became more familiar with the technology, we encountered further limitations and made additional compromises. These are described below.

First, we found that the ACMS language was not able to handle the level of programming logic required in our client application; it is geared towards, and is very good at, the routing of work between servers and terminal users. Consequently, parts of the client application were coded in COBOL and bundled into the ACMS servers along with the business server modules. We called these *client procedures*, and they were required to obey all the rules for a client application: no direct access to the database, and all work done through the real business servers.

ACMS requires that its servers be packaged a certain way for use in this environment, and this means they are *not* accessible to other non-ACMS client applications. We could re-use the business server modules, but we were required to re-package them for use elsewhere.

We found that trying to implement each business server as its own ACMS procedure server introduced excessive overhead and complexity into the system. In addition, one business server could not gain access to a different one unless they were bundled into the same ACMS server, so there was limited code sharing that could take place. Our solution was to bundle related business servers into the same ACMS server, along with the client procedures that use them. This allowed for better re-use of common subroutines. Also, we were able to use fewer server processes, since one process could provide multiple services and stay busy more of the time.

We ended up with three types of ACMS server, one controlling general access to the system, one performing pure registration functions such as adding and dropping classes, and one providing class information. Multiple instances of each type are run, based on frequency of usage. This has provided acceptable response on a fully loaded system.

At the presentation interface, we found that ACMS could not talk directly to the voice processing hardware, and the voice hardware did not connect directly to the network. Consequently, additional modules were implemented to facilitate the communication between the client procedure in ACMS and the voice server. We insisted that no new functionality be introduced here: the new modules were to be transparent as far as content of the messages was concerned, and could act only to move messages to the correct destination. We consider these modules to be middleware, and we expect to replace them as better functionality becomes available in ACMS and on the IVRU.

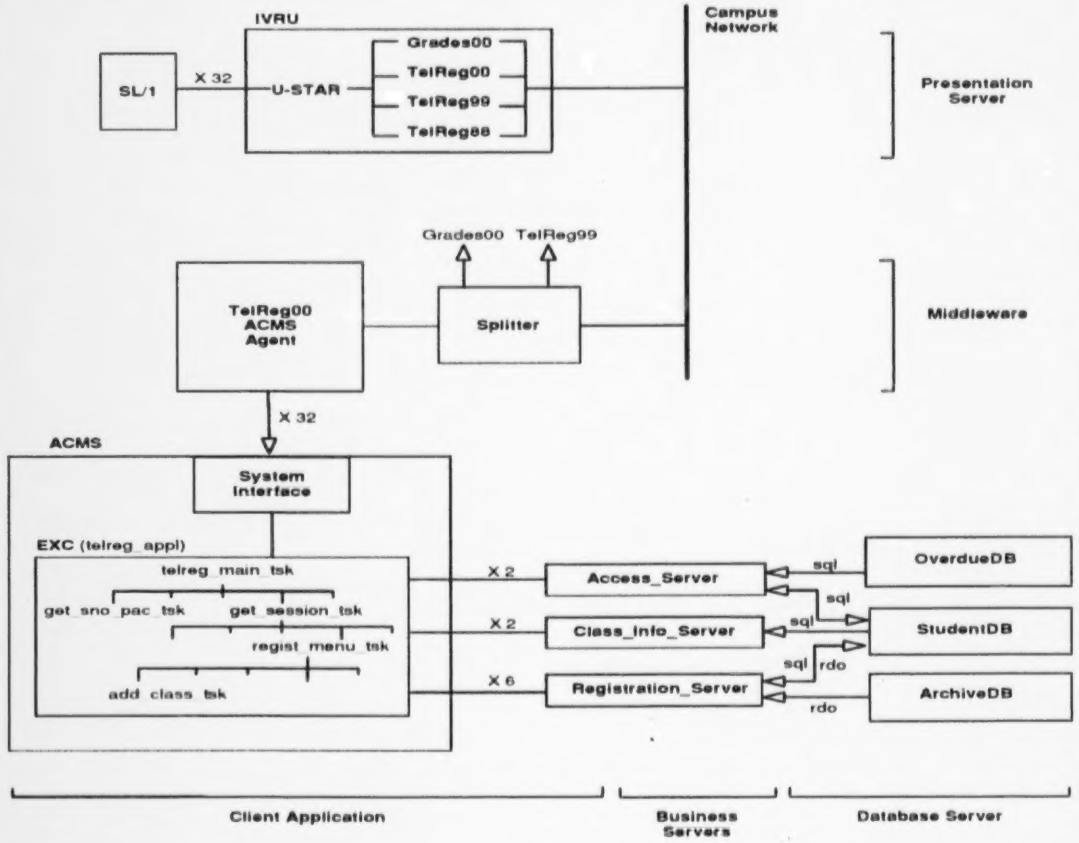

IVRU
SL/1
X 32
U-STAR
Grades00
TelReg00
TelReg99
TelReg88
Campus Network
Presentation Server
Grades00
TelReg99
TelReg00 ACMS Agent
Splitter
Middleware
X 32
ACMS
System Interface
EXC (telreg_appl)
telreg_main_tsk
get_sno_pac_tsk
get_session_tsk
regist_menu_tsk
add_class_tsk
X 2
X 2
X 6
Access_Server
Class_Info_Server
Registration_Server
sql
sql
sql
sql
rdo
rdo
OverdueDB
StudentDB
ArchiveDB
Client Application
Business Servers
Database Server

Figure 5 : Telephone Registration Architecture

Figure 5 shows our architecture, with notations as to which software modules correspond to each of the four boxes. Note that the U-STAR front-end can route calls to other applications, and that these are tapped off at the middleware component labelled *Splitter*. The inner structure of the three ACMS procedure servers is not shown, but has a well-defined dividing line between modules that are part of the client application and ones that are part of the business servers.

In summary, the fact that ACMS is a transaction processing environment rather than a client/server environment has pushed our implementation away from the 4-box architecture. Even so, all the compromises we made were made consciously, and the spirit of the architecture was always followed. We have ended up with a set of software modules that have clearly defined roles, that localise functionality, and that only need to be re-packaged to operate in a different environment.

The Future

As hardware becomes more powerful, database management systems will become more sophisticated, and allow incorporation of more business rules into the database itself. Examples of this that are becoming available today are field and record validation rules, triggers, enforced referential integrity rules, and stored procedures. Unfortunately, these are not yet sophisticated enough to handle all the processing that we need in our business function servers. But we expect that as these and other features mature, there will be less need for business servers, and they will eventually be subsumed by the DBMS. Then, the four boxes reduce to just three, the distinction between operational and decision support systems disappears, and things look much more like the PC community views them today.

Bibliography

Inmon, W.H. Developing Client/Server Applications in an Architected Environment. Boston: QED Technical Publishing Group, 1991

Perlitch, Mitchell D. Client/Server Computing: Towards an Integrated, Network Computing Environment. Digital Equipment Corp, 1990

Sudama, Ram Distributed Application Methods: Nine Approaches to Communications and Information Sharing. Digital Equipment Corp, 1991

"The more things change..." Computerworld, 4 May 1992, pp. 83-84

SOLAR
Harvard's Client/Server Based Fundraising Management System

James Conway, Director, Development Computing Services
Philip Gow, Associate Director, Distributed Systems
Mary Reaney, Sr. Programmer Analyst
Harvard University
Cambridge, MA 02138

Abstract

During the planning phase for its multi-billion-dollar fundraising campaign, Harvard University recognized the importance of information technology to the campaign's success. The University is implementing a fundraising management system on a relational database, accessed by desktop systems using client/server technology. The project, known as SOLAR, is an interactive fundraising system designed to provide fundraising management and personnel the capability to access and share management summary and prospect information, such as automatic summary charts, prospect tracking, research, events and so forth.

University management is able to view, from the desktop, up-to-date campaign status information in the form of charts from a wide variety of views such as goal versus actual charts by University, School, theme, program, fundraiser, class, geographical area, etc. These charts may be selected by a wide variety of possible views. For example, the President may desire to see how a school is progressing within the Health theme, while the Director of Development may need to analyze how a class is proceeding with the technology theme. These charts are also available in reports that may be viewed on the desktop display and/or printed to a report. All charts may be cut and pasted to wordprocessing documents, spreadsheets, or presentations.

Introduction/Strategy

Although possessing a well-invested endowment, Harvard University is facing many challenges when it comes to providing the best possible educational environment. Maintaining leading-edge laboratories in the modern fast-moving high technology fields, for example, requires large annual allocations of limited funds. Salaries and benefits for outstanding faculty and staff continue to rise as does the ever-increasing burden of facilities maintenance. In order to meet these challenges and prepare for the next century, Harvard will be initiating a multi-billion dollar campaign.

Currently, Harvard processes over $200 million in gifts annually with a COBOL-based application on a pair of Hewlett-Packard minicomputers. Like many organizations, management was reluctant to invest additional funds on a new and, as far as the University was concerned, untried technology. In short, in order to meet the long-term objectives of the University and the demands of the upcoming campaign, it was decided that a two-staged strategy would be followed. The first stage retains the investment in the existing transaction system while moving forward with client-server technology to support the ambitious fundraising effort. The second stage calls for the replacement of the aging transaction system with transactions being processed through client-server technology.

By following this strategy, management is not completely committed to a technology until it has been proven. Although the second stage depends on the completion of the first, the strategy may be limited to the first stage and still obtain significant improvement in information services.

The first stage of the strategy is based upon the concept of following a plan that *extends* the legacy of the COBOL system beyond transaction processing to a client-server environment in which the fundraising information is readily available in a form that can be integrated into the fundraiser's overall scheme for displaying and analyzing information. While the existing system continues to process transactions, emphasis will shift increasingly toward extension. By following this strategy, Harvard will be able to preserve its investment in its existing system and current desktop computers while giving administrators, development, and alumni support personnel easy and immediate access to alumni, prospect, and gift information through their microcomputers. Through the use of the simple point-and-click method, information will be available, directly from the server, in report, on-line, or chart form.

Once the first stage of the strategy is in place, the processing model will appear as depicted in figure 1. The legacy system will continue to process gift and biographical information. As the transactions are applied to the old data base, information will be transferred to the appropriate relational data base located on the Sun 690MP data-base server. The Sybase relational data base will provide secure, efficient access to up-to-date biographical and gift information. In effect, this separates the day-to-day gift and biographical transaction process from the more ad-hoc inquiry and reporting process. This separation of the data-gathering function from the data-access function will lead to a more efficient use of the legacy system, while at the same time improving reporting. For example, fundraisers would be able to generate their own labels and reports locally rather than following the current time-consuming method of requesting reports from the central MIS group.

Unlike many warehouse systems, SOLAR is not designed for data access only. It is designed to be a system that uses the strengths of warehouse type systems while adding the utility of interactive systems. SOLAR contains two categories of information: core and fundraiser-specific. Core information consists of those pieces of information that are entered and maintained by the legacy transaction system. Core transaction information consists of data such as individual gifts, pledges, and matching gifts. Fundraiser information is relevant only to the fundraising function and is added to and maintained directly by the SOLAR system. Fundraisers work with prospects in a fashion similar to sales and marketing personnel in the business world: rating the prospect's ability to give, researching the prospect's background and interest, developing cultivation activities and events, and, of course, completing the actual ask.

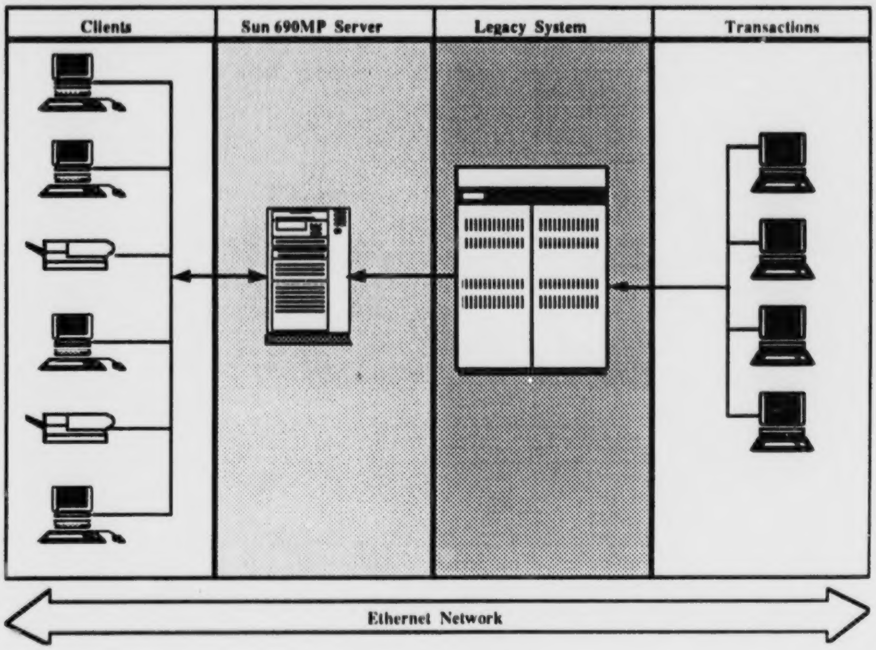


Figure 1 Phase One

Once the first stage has been completed, migration to the new environment may commence (or at management's discretion, the project may be halted without disruption in processing). Migration has as its objective the replacement of the legacy system with transaction processing moving from the old system to the client-server model. During this phase, maintenance of alumni and prospect core information will be performed on desktop computers connected to either a central server as depicted in figure 2 or to individual servers located at each of the schools. Although analysis of this phase is just beginning and many decisions are yet to be made, preliminary plans call for an in-depth evaluation of using Sybase's Replication Server to distribute not only functionality but also the information. Under this model, the information pertinent to the individual school will be located at the school and yet be available for access from any of the other schools over the University network.

Project Organization

To be successful, it was essential that an organizational structure, with its accompanying responsibility, be defined and agreed upon by all parties. Since SOLAR's objective is to construct a system that fulfills functional needs, management not only needed to be involved but committed to its success. With this in mind, the project organization shown in figure 3 was proposed to senior management and accepted to manage the project. It brings to bear a wide range of management and program experience while providing necessary commitment.

The Project Sponsor, at the Executive Director level and possessing the primary line authority for the project upon its completion, has overall authority for the project. As the Senior Staff manager that funds the project, the sponsor is responsible for the project's scope and overall quality.

The *Project Manager* is a full-time employee reporting to the Information Technology Director, is the leader of the project, and is full time for the life of the project. The project manager is responsible for developing estimates and schedules, preparing funding recommendations, managing the project team and communicating project status to the Project Sponsor and Steering Committee.

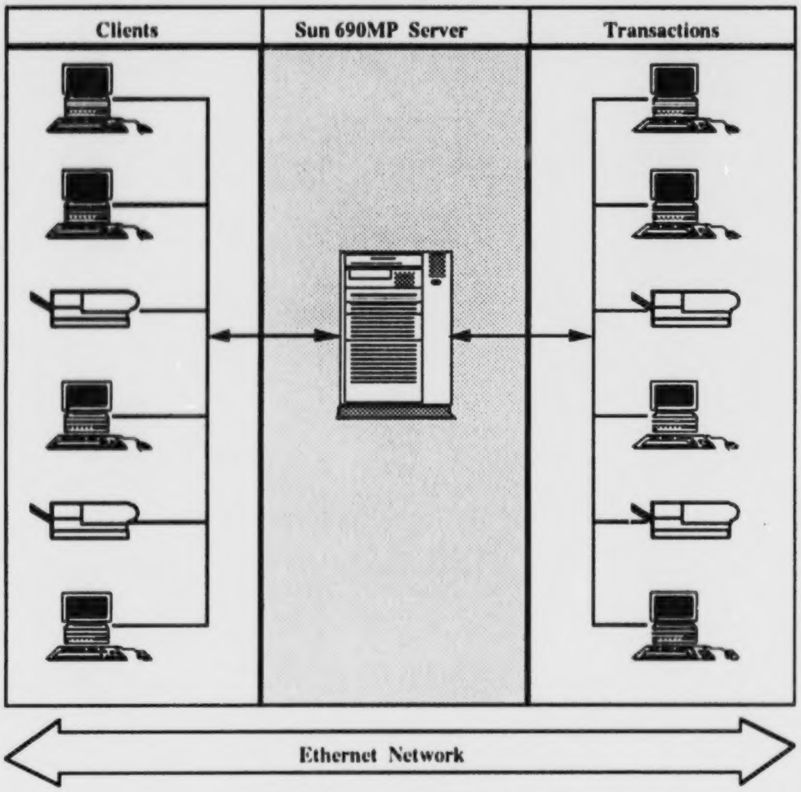


Figure 2 Phase Two

The *Quality Review Committee* consists of senior staff and/or supervisors from the departments affected by the project. The committee reviews the project at specified milestones and is responsible for the project's meeting its objectives in an efficient and effective manner. The committee ensures that the project solves the right problems and ensures the quality of the finished product.

The *User Coordinator* ensures that staff members from the interested departments are available, as needed, during the life of the project. The user coordinator should be at the manager level and have direct reportability to either the Vice President or the Executive Director. Coordinating all user tasks, the user coordinator keeps the Project Sponsor informed of all problems that may arise in the allocation of user resources and chairs the Quality Review Committee.

The *Project Team*, reporting to the project manager, consists of members of both the IT department and user departments. Team members carry out the project tasks and are responsible for the overall design, construction and implementation of the project. Computer projects require a wide variety of skills that include analysis, systems design, programming, training, and operations.

While the team organization needed to be flexible, in that the project required more resources during the middle phases than at the beginning and end, the management structure remained permanent.

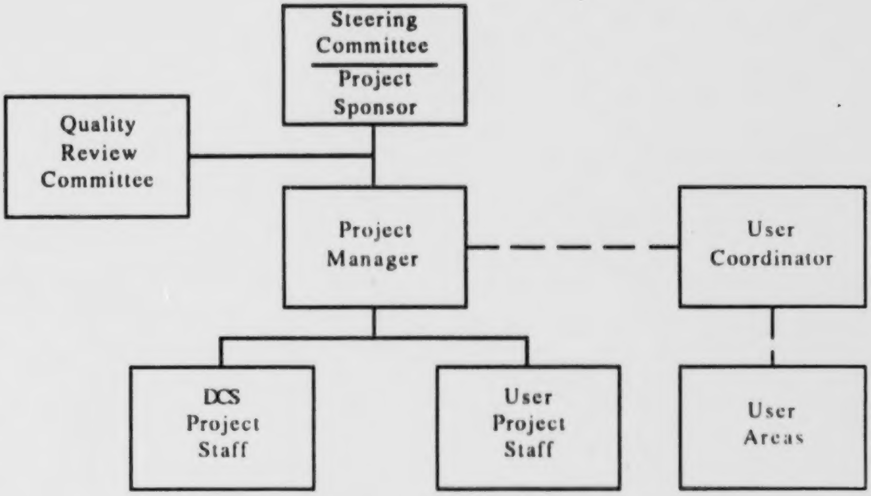


Figure 3 Project Organization

The SOLAR System

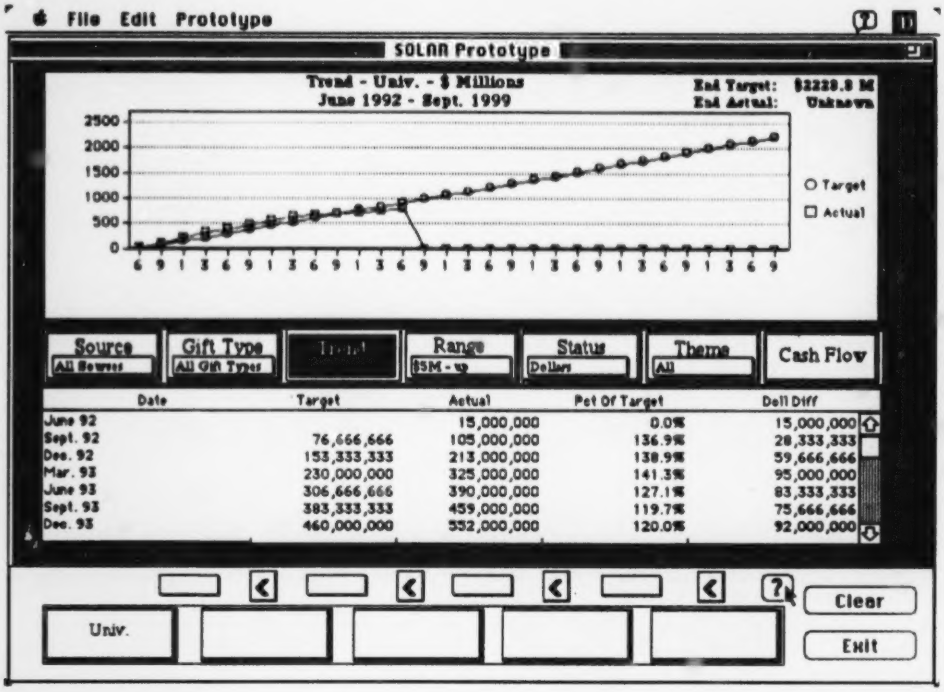
SOLAR, scheduled to go live in early 1994, is a client-server fundraising system designed to provide management and fundraising personnel the capability to access and share summary and prospect information. The server, storing information on over 500,000 prospects and nearly two million individual gifts, is a Sun 690MP utilizing the Sybase relational database. The Sun 690, with four coprocessors, is configured with 256 megabytes of memory and 16 gigabytes of disk capacity. The clients consist of nearly 200 Apple 610s and 650s configured with 20 megabytes of memory and 230 megabytes of disk. The client front-end application software is ACI's Fourth Dimension and provides access to the Sybase database over an ethernet network. Similar to many sales and marketing systems, SOLAR will provide a wide-range of functions that include:

- management charts
- ad-hoc report writer
- prospect management
    - prospect tracking
    - research summaries
    - prospect clearance
    - gift and gift history tracking
    - planned giving
    - volunteer and committee activities and membership
    - event management
- merge selection of prospects, singly or in groups, to word-processing documents
- data extracts

Management Summaries

SOLAR management summaries are organized to provide important fundraising progress information in management chart and text formats directly to the microcomputer display. University management will be able to view, from the desktop, up-to-date campaign status in the form of charts from a wide variety of views such as goal versus actual charts by University, School, theme, program, fundraiser, class, geographical area, etc. These summaries are organized to meet the differing information needs found in the management hierarchy. At the highest level of summarization, SOLAR displays University-wide views of campaign progress. In addition, management is capable of "drilling down" the summarization hierarchy to more levels of detail. For example, SOLAR provides management the capability to review progress for Themes for the University as a whole or, by simply clicking on another selection, to review an individual school's overall progress.

One of the summarizations that management finds helpful in its planning efforts is a chart of the campaign's trend (figure 4). To view the campaign's trend from a University perspective, the manager


File
Edit
Prototype
SOLAR Prototype
Trend - Univ. - $ Millions
June 1992 - Sept. 1999
End Target: $2228.8 M
End Actual: Unknown
2500
2000
1500
1000
500
0
6 9 1 3 6 9 1 3 6 9 1 3 6 9 1 3 6 9 1 3 6 9 1 3 6 9 1 3 6 9
Target
Actual
Source
All Sources
Gift Type
All Gift Types
Trend
Range
$5M - up
Status
Dollars
Theme
All
Cash Flow
?
Clear
Exit
Univ.

Date	Target	Actual	Pct Of Target	Doll Diff
June 92		15,000,000	0.0%	15,000,000
Sept. 92	76,666,666	105,000,000	136.9%	28,333,333
Dec. 92	153,333,333	213,000,000	138.9%	59,666,666
Mar. 93	230,000,000	325,000,000	141.3%	95,000,000
June 93	306,666,666	390,000,000	127.1%	83,333,333
Sept. 93	383,333,333	459,000,000	119.7%	75,666,666
Dec. 93	460,000,000	552,000,000	120.0%	92,000,000

Figure 4 High Level Trend

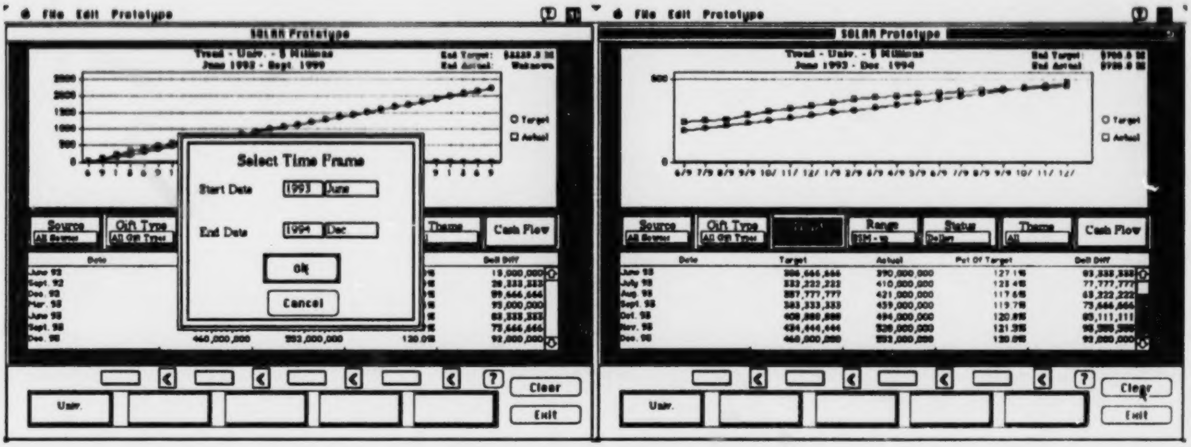
simply selects "University" from the hierarchy buttons (located on the lower section of the screen) and clicks on the "Trend" button. The lower half of the Trend Screen displays, by quarter, the target amount to be raised up to that date, the actual amount raised to that date, percent of target and the amount of difference. To the right of these figures is located a "scroll" bar that will allow the manager to view the trend for all the campaign years by quarter. The upper "window" shows a graph of actual monies raised (outright gifts plus pledges) vs. planned objectives based on the figures shown in the lower "window".

Although this high level view of the campaign's trend may be useful for an overall view, it doesn't provide a fine enough scale to view trends in more specific time frames. To meet this need to view smaller ranges in time, SOLAR provides a mechanism to view trend in "slices of time". For example, rather than viewing trend for the entire campaign, if management would like to see the trend for the specific date range of June 1993 to December 1994, they click on the "?" button. SOLAR returns a dialog box, as shown in figure 5, asking the user to enter the desired date range. The user enters the

"Start Date" year and month and the "End Date" year and month by simply selecting the appropriate years and months from the pop-up buttons and clicking on the "OK" button.

Similar to the high-level view, the lower half of the Trend Screen displays, for the selected date range, the target amount to be raised, the actual amount raised, percent of target and the amount of difference. The upper "window" shows a graph of actual monies raised (outright gifts plus pledges) vs. planned objectives based on the figures shown in the lower "window".

The example given is but a brief view of the type of graphing available in SOLAR. Fundraising management is able to drill down and, at the same time, graphically view the information a number of different ways. As you can see from figure 4, there are seven major types of graphs available to the fundraiser. In addition, each of the seven major groups contains a number of chart/display options. The introduction of automatic and up-to-date management charts is expected to not only increase management efficiency but also, just as importantly, to free management and staff time from the time-consuming efforts required to build similar charts by hand.



Select Time Frame

Start Date 1993 June

End Date 1994 Dec

Figure 5 Trend Range Selection

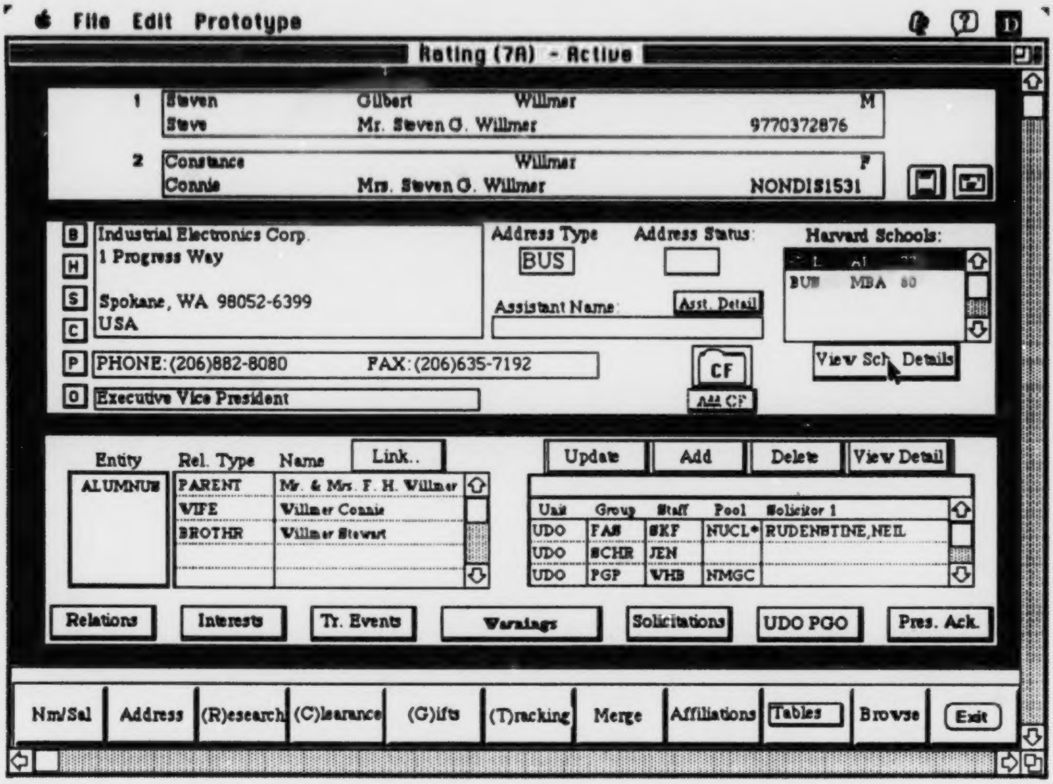
Reporting

Similar to many other applications, report generation is a critical piece of the design. With over 40 local laser printers spread throughout the organization, fundraisers will be able to select preprogrammed reports directly from their desktop, view the report on their screens before printing, and print the selected report directly to their local laser printer. When a one-time report is needed, the system will include a powerful report writing facility. Similar to the functionality for reports, SOLAR will provide users the choice of using their personal standard extracts via the push of a button or the ability to execute one-time ad-hoc extracts. In addition, the user will be able to select information from the database and automatically link to other client application packages such as Microsoft's Word.

The benefits of introducing the ability to select and print reports at the local laser printer are obvious: report turn-around has been reduced to minutes; the actual volume of paper used is reduced (the users select only what they want to see - not an entire database dump); users are able to select what is to appear on the report and the sort sequence that they, as individuals, wish to see.

Prospect Management

The main objective of SOLAR, of course, is to provide a tool for the fundraiser to manage information on his or her prospects and to share that information with other fundraisers on a University-wide basis. The project team decided that the best way to meet the needs of the fundraiser for quick and easy access to the information was through the use of a graphical interface using a simple point-and-click method. Using the graphical interface, a fundraiser obtains all relevant information on a prospect by simply entering the pertinent search information such as the prospect's name. SOLAR returns a display as shown in figure 6.


File
Edit
Prototype
Rating (7A) - Active
B
H
S
C
P
O
Address Type
Address Status:
Harvard Schools:
Assistant Name:
Asst. Detail
View Sch Details
CF
Add CF
Entity
Rel. Type
Name
Link..
Update
Add
Delete
View Detail
Relations
Interests
Tr. Events
Warnings
Solicitations
UDO PGO
Pres. Ack.
Nm/Sal
Address
(R)esearch
(C)learance
(G)ifts
(T)racking
Merge
Affiliations
Tables
Browse
Exit

1 Steven Gilbert Willmer M
Steve Mr. Steven G. Willmer 9770372876

2 Constance Willmer F
Connie Mrs. Steven G. Willmer NONDIS1531

Industrial Electronics Corp.
1 Progress Way

Spokane, WA 98052-6399
USA

BUS

BUS MBA 80

PHONE: (206)882-8080 FAX: (206)635-7192

Executive Vice President

ALUMNUS

Rel. Type	Name
PARENT	Mr. & Mrs. F. H. Willmer
WIFE	Willmer Connie
BROTHR	Willmer Stewart

Unit	Group	Staff	Pool	Solicitor 1
UDO	FAS	SKF	NUCL*	RUDENSTINE,NEIL
UDO	SCHR	JEN		
UDO	PGP	WHB	NMGC	

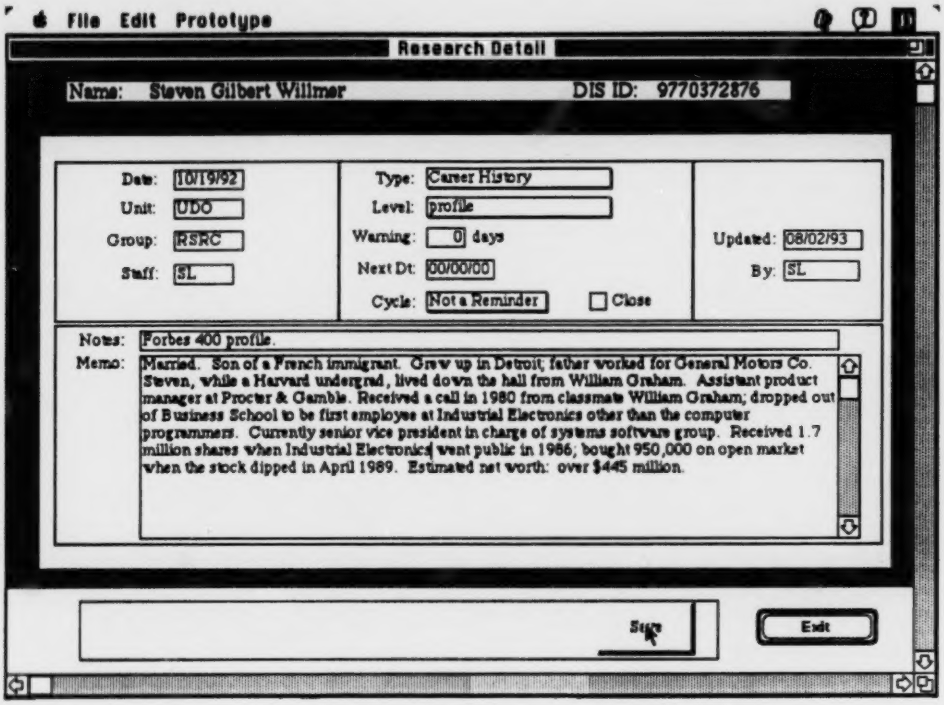
Figure 6 Main Prospect Screen

Top section of the screen displays the prospect's giving rating along with the prospect's name and identification number. The spouse's name and ID number are displayed directly below the prospect's name. The second section of the screen displays the prospect's address information. The prospect's preferred address is displayed on the screen initially. To display other addresses, the user simply clicks on the buttons "B", "H", or "S" to display the business, home, or seasonal addresses. For example, the fundraiser clicks on the "C" button to display the current address, the "P" button to display the preferred address, or the "O" button to display any other address. The selected address then appears in the address area, leaving all the other information as originally displayed.

Along with address information, the second section contains information about the prospect's assistant and related school information on Harvard degrees. The fundraiser may see the schools, degrees and years by scrolling in the Harvard Schools box. If he or she double clicks on a school record, the School Detail Screen will appear, displaying information for the selected school record such as degree awarded by Harvard, years of attendance, concentration, honors, and house of residence.

The third section displays the names of people related to the prospect and the relationship: wife, husband, son and so on. Next to relationships is located summary information on solicitations related to the selected prospect. The view areas for relationships and solicitations may be expanded by simply clicking on the appropriate button.

In addition to those mentioned above, buttons are provided to allow the user to access more information on a given area. The buttons located on the bottom of the Main Prospect Screen, for example, allow the fundraiser to access and/or enter more in-depth information on the prospect such as Name and Salutations, Address, Research, Gift and Giving histories, and Tracking information. Figure 7 shows an example of research information on a prospect that may be accessed by simply clicking on the Research button.



Name: Steven Gilbert Willmer
DIS ID: 9770372876

Date: 10/19/92
Unit: UDO
Group: RSRC
Staff: SL
Type: Career History
Level: profile
Warning: 0 days
Next Dt: 00/00/00
Cycle: Not a Reminder
☐ Close
Updated: 08/02/93
By: SL

Notes: Forbes 400 profile.

Memo: Married. Son of a French immigrant. Grew up in Detroit; father worked for General Motors Co. Steven, while a Harvard undergrad, lived down the hall from William Graham. Assistant product manager at Procter & Gamble. Received a call in 1980 from classmate William Graham; dropped out of Business School to be first employee at Industrial Electronics other than the computer programmers. Currently senior vice president in charge of systems software group. Received 1.7 million shares when Industrial Electronics went public in 1986; bought 950,000 on open market when the stock dipped in April 1989. Estimated net worth: over $445 million.

Figure 7 Research Screen

One of the most popular SOLAR features that will be available to the fundraiser is the Tracking function. During a cultivation of a prospect, many individual events occur that are important to note and/or share with other fundraisers interested in the prospect. The fundraiser, for example, may have just returned from a prospect luncheon meeting and have collected relevant bits of information on a prospect's interests. Upon returning to the office, the fundraiser simply accesses the prospect's individual record via the Apple Centris, presses the Tracking Button and enters a comment on the luncheon and dates the comment for further reference. If the fundraiser learns that the prospect's daughter is planning to be married two months in the future, the fundraiser enters a "Tickler" to remind him/her to send a note to the prospect 10 days before the wedding by simply entering the reminder as a comment and entering that he/she would like to be reminded of this event 10 days before a specific date. The system, 10 days before the specified date, will automatically notify the fundraiser to send the note.

Another feature of SOLAR, Merge facility, allows the fundraiser to build a wide variety of name and address merge queues that may be automatically merged to any word-processing document. For example, a fundraiser may plan to hold a number of prospect-review meetings at various locations throughout the country. Each session will be held with a different group of volunteers and, thus, will require different sets of correspondence depending on the meeting topic and attendees. Through the merge facility, the fundraiser builds a list of attendees and their addresses for each meeting. As the events are held throughout the year, the fundraiser simply selects the event by queue name; the current names and addresses meeting the original selection criteria are selected and merged to the designated word-processing document.

Critical Management Issues

This case illustrates an important point about the strategic use of information technology in the support of a university's overall objectives. Although the system and the technology upon which it is built are important to a school's success in achieving and maintaining the funds required to provide a first-rate educational environment, an overall fundraising strategy developed by university management that leverages technology must be in place. A committed, informed executive management that supports the use of information technology is a prerequisite for the successful development and installation of such a system. Management must know the needs of the school and possess an understanding of information technology. In addition, a member of the executive management team, if not the President, must also be willing to champion the system and create the environment in which information technology and its use are considered important to the school's success.

In addition to this high-level support, it is important to identify a project sponsor. As mentioned earlier, the sponsor is a high-level manager that has overall responsibility for the project. Ideally, the project sponsor should be the manager responsible for the primary function supported by the system. For SOLAR, this meant that the manager responsible for the overall management of the fundraising campaign has responsibility for not only fund-raising activities but the supporting information system as well.

Although management support and involvement is critical to the success of any information technology project, buy-in from the people that are actually expected to use the system is just as critical. Nothing causes a system to fail more than the underestimation of the importance of the people in the trenches and their agreement to use the system. If these people feel a system is being forced on them without first soliciting their input and allowing them to critique it in a meaningful way, the system will not be used to its full potential or, in the worst case, the system will not be used at all. To achieve buy-in, the SOLAR team consisted of a number of fundraisers charged with the responsibility to develop, with the aid of computer specialists, a full prototype of the system. Once the prototype met the fundraising team's specifications, in-depth reviews of the prototype (each review session took one to three days) were provided for the entire 200 person user base. Problems and requested changes raised in each session were applied to the prototype and ready for the next review, showing the users not only that their input was listened to but that their requested changes were regarded by the team as important. This approach, although time consuming, raised overall morale, the desire for the new system, and eased the trauma commonly associated with change.

Summary and Conclusion

Four factors were critical to the success of SOLAR. First, SOLAR was initiated and championed by responsive and visionary management. Management understood how the application of information technology would be an important ingredient in the overall success of the fundraising campaign. Second, fundraising management was ready for change and willing to encourage other departments such as gift processing to participate in the new system. The third factor was the selection of the project leader. The Project Leader needed not only to understand the new technology and its application, but also to possess an in-depth understanding of fundraising. Fourth was the overall method of communication and involvement. From the start, the project involved not only computer personnel but fundraisers as well. Management appointed a User Coordinator to work with the team at least 50% of his time. In addition, buy-in was achieved by allowing all fundraising people to review and critique the system prototype during development.

The benefits we expect to reap with SOLAR are many, but chief among them are:

- it offers fundraising personnel a quicker and more easily used mechanism for accessing and reporting data, since a client/server architecture is able to respond to queries more completely than is possible when data is transmitted in a terminal-oriented system, one screenful of data at a time;

- the officers have access to current information, since queries are accessed in "real time" rather than against data that resides in a file that has been previously downloaded;

- the amount of effort required in training new staff is expected to drop sharply due to the ease of system use;

- the amount of time that it currently takes the computer staff to develop programs for various functions and reports will be reduced by a factor of 10 to 20 times.

The ability to access and enter fundraising information through simple, easy-to-use methods means that fundraising personnel will have information at their fingertips when and where they need it and will not have to wait for the computer staff to write reports and/or extracts. The fundraiser, through this direct access to critical information such as giving histories, is able to answer donor queries while still on the telephone - thus improving personal relationships between the fundraiser and the donor.

We anticipate that SOLAR will contribute to a fund-raising success unequaled in the history of fundraising in higher education. Working with over 500,000 prospects, the University will raise two billion dollars over the five-year life of the campaign. These funds will allow the University to maintain and increase its ability to provide a first-rate education for its impressive student body into the next century.

Talking Turkey About "Real Change"

Carole Cotton
CCA Consulting Inc.
Wellesley
MA 02181

ABSTRACT

We see a coincidence of new management and new computing models, with information being viewed as a prime enabler in meeting institutional goals. The prognosis is for more change -- not less.

What is the nature and timing of this change? What business needs are driving IT changes? Are organizational structures being affected? What are the motivations behind the move to client-server? How fast is the movement and where are the early adopters? Who and what is connected to the campus backbone and what services are provided? What technologies and issues do IT professionals see as most important to the future of their own institutions?

This paper addresses these questions by reporting on the answers given directly by the institutions themselves and represents a documented snapshot of "real" change over the past three years. The data for the paper are derived from our survey of Higher Education conducted during the spring of 1993, combined with data from surveys conducted the previous two years.

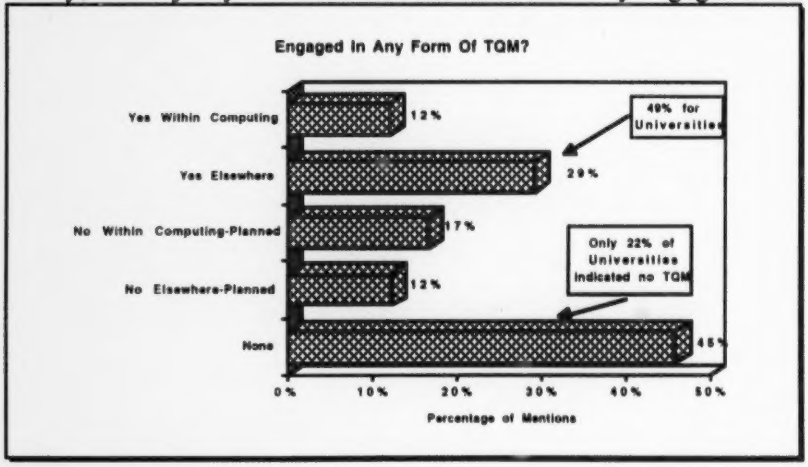
Introduction

Our 1991, 1992 and 1993 studies entitled: Understanding Information Systems in Higher Education reported significant change in many aspects of IT on campus. While, it is not possible, in this short paper, to summarize three 200 page reports, we will discuss several areas of pivotal change.

The data for this paper are derived from our survey of Higher Education conducted during the spring of 1993, combined with data from surveys conducted the previous two years, The text is further augmented by comments obtained by interviewing many notable Higher Education trendsetters. We begin at the top by looking at key management issues.

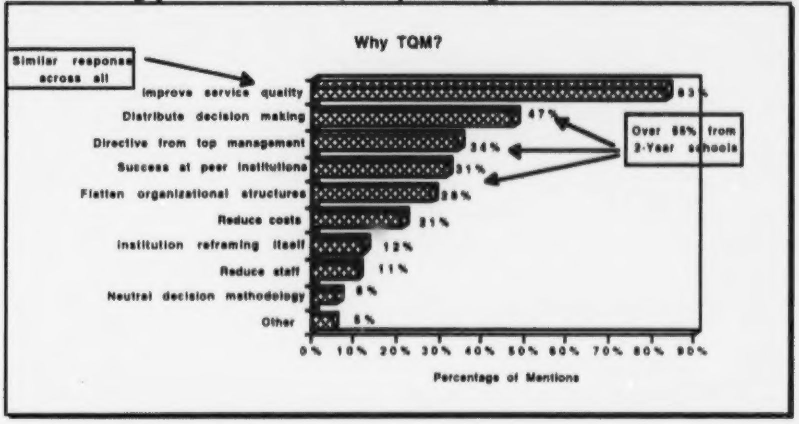
The Challenge of Re-Framing Institutions

Today, the majority of institutions are either currently engaged in -- or planning to -- undertake some form of re-engineering with the dual goals of improving overall institutional effectiveness and controlling costs. We asked several questions about institutional activities in TQM, and found that 41% are currently engaged in some form of TQM. Further, 22% of those who reported no current TQM activities are planning for it.



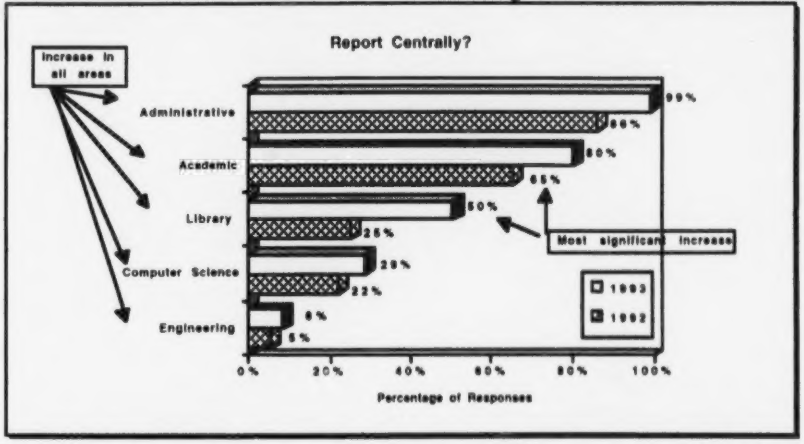
As expected, the Universities tend to be ahead of all other classes of institutions. Only one in five Universities reported that had **no** TQM activities. In comparison, the total market ratio is just under one in two. Twelve percent of all respondents (30% of all Universities) indicated TQM activities within Computing and an additional 17% that they were planning for it. (Please note that this was a multiple response question and therefore, responses do not add to 100%.)

Motivations for Undertaking TQM



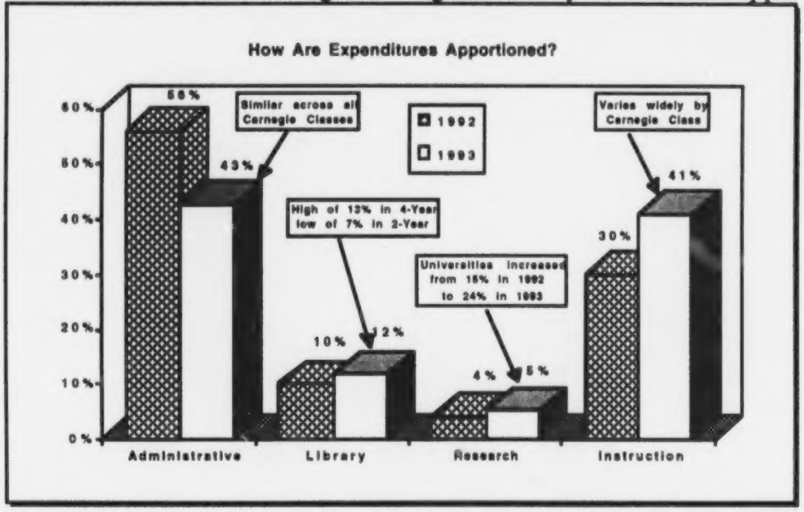
The starting point in Total Quality Management is to focus on the customer, and it appears that Higher Education is doing just that! Looking now at the motivations for undertaking TQM activities, the most frequently mentioned reason (83%) was to "Improve Service Quality," and the pattern is consistent for all Carnegie classes. Oddly, only 21% mentioned "Reduce Costs" as a motivator. There were, however, different responses across the Carnegie classes; the Universities (30%) and the 2-Year (32%) schools were both above the total market response. And even fewer (11%) mentioned "Reduce Staff" as a reason for undertaking TQM. The Carnegie class responses to this question also varied widely. Two-Year schools did lead in this area, with 28% of mentions, while 4-Year schools, with just 2% of mentions, anchored the bottom end of the curve.

Which Computing Organizations Report Centrally?



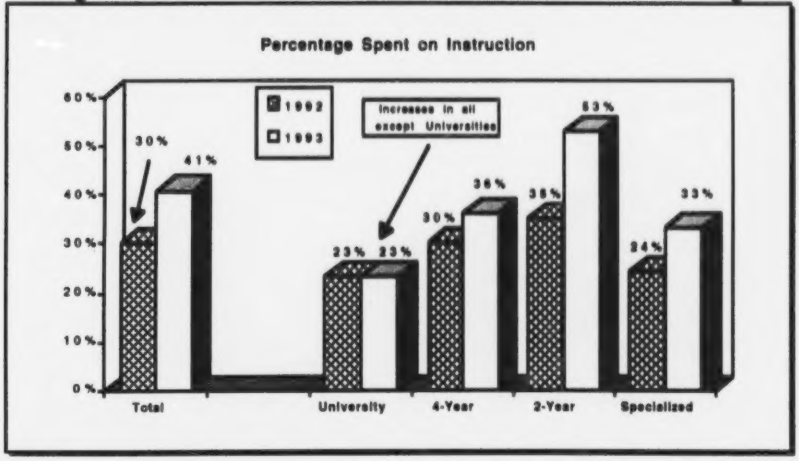
Does the move to distribute decision making and therefore access to information result in more centralization of computing organizations. Looking at two years of data, we see what appears to be a genuine movement to consolidate computing organizations. When asked which computing organizations report centrally, our respondents indicated increases in all areas. WHY? Perhaps because the move to distributed computing creates a compelling need to build and maintain a coherent IT infrastructure in order to support decentralized information needs.

How Are Computer Expenditures Apportioned?



What effect are these changes having on the ways institutions apportion their total -- not just central -- expenditures for computer hardware and software across the major functional areas? And, has it changed over the last year? Looking first at the overall picture, we see that Administration is down while Library, Research and Instruction are all up. There are significant pattern differences among the Carnegie classes. Only Administration, at 43%, is similar across all Carnegie classes. Library varies from a high of 13% in 4-Year schools to a low of 7% in 2-Year schools. Research expenditures, as a percentage of the Total, are highest in Universities, where this percentage increased from 15% in 1992 to 24% in 1993 and lowest (<2%) in 2-Year schools. The big story is in Instruction.

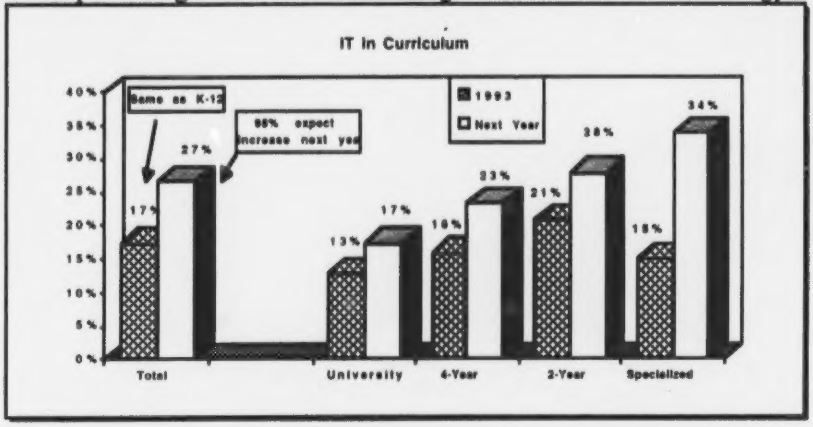
A Closer Look at Instruction



Taking a closer look at Instruction, we see that across all of Higher Education, 41% of total expenditures on computer hardware and software were devoted to this area in 1993. This represents a 37% increase over last year. Further, we saw increases for all Carnegie classes -- except for Universities -- where it *appears* that funding stayed flat. Did the Universities actually "flat fund" Instruction, or are we seeing distorted responses resulting from the fluid boundaries which define Instruction and Research? The most significant change is in the 2-Year schools who reported that they spend 53% of their Total computing expenditures on Instruction. Is there a correlation between the percentage of funding allocated to Instruction and the rate of IT integration into the curriculum? They do appear to be correlated.

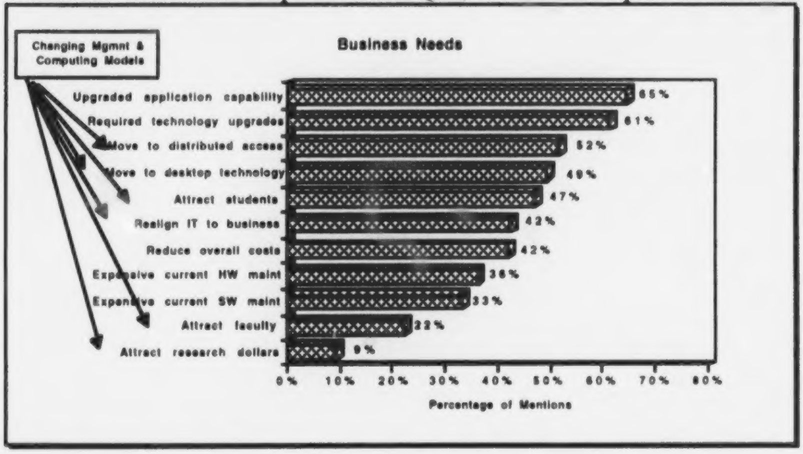
How Pervasive Is IT In The Curriculum?

What percentage of courses have integrated information technology into the curriculum? By "integrated," we mean that *IT is used an integral component in the course, as differentiated from simply using word processing to type papers.* Overall, the mean response was 17%. Coincidentally, this is the same mean response we found in our K-12 study! And the best news is that 98% of our respondents are projecting an increase next year. The mean percentage of courses which will integrate IT into the curriculum is projected to be 27% next year -- reflecting increases across all Carnegie classes -- but once again we see large differences, which range from a low of 17% in Universities to a high of 34% in Specialized institutions. (Please note that we have taken all responses at face value. It is likely that there are wide variances in respondents' interpretation of what is meant by "technology integration into the curriculum.")

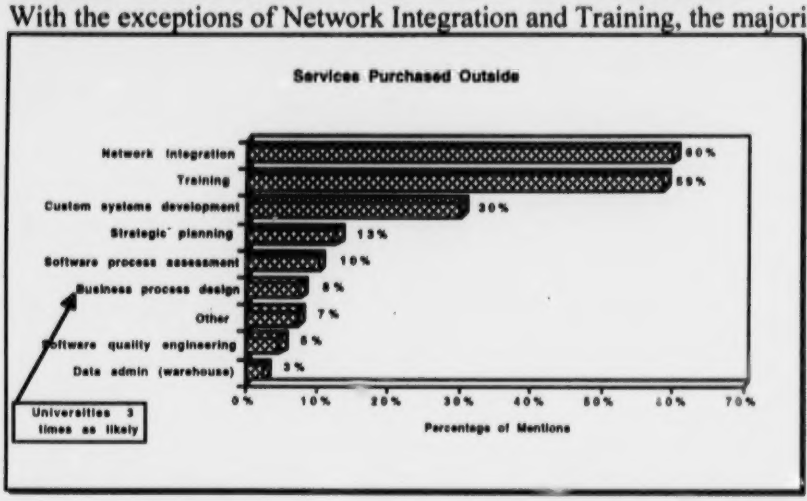


What "Business" Needs Drive IT Expenditures?

Given the increased emphasis on TQM, we would expect to find a close alignment between the mission of the institution and IT. When asked to define which business needs drive technology expenditures, we found that those needs relating to changes in computing models and/or management models received a high percentage of the responses. Further, the top two mentions: Upgraded Application Capability and Required Technology Upgrades are likely to be the resulting tactical changes required to implement new management and new computing models. Looking beneath some of the market-wide responses, we found that the University responses were significantly above the market average in *most* areas, while the other classes tend to cluster about the mean.

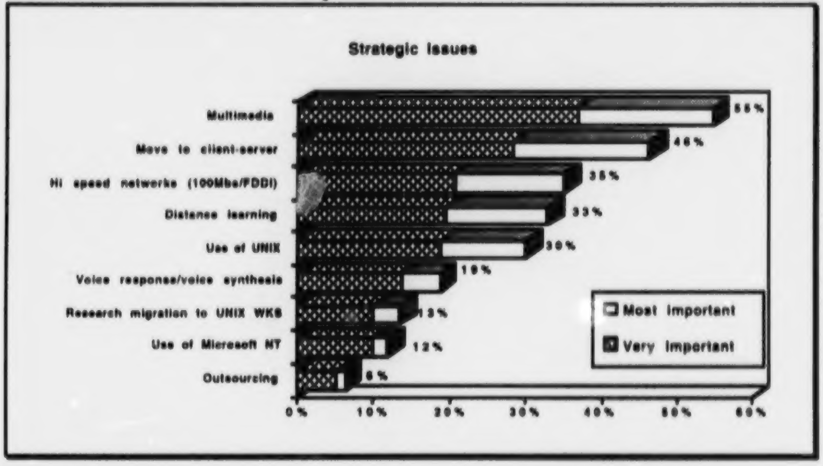


What Services Will Be Purchased?



With the exceptions of Network Integration and Training, the majority of institutions did not indicate that they would be purchasing large quantities of outside services. Except for the Universities, the responses for the Carnegie classes are consistent. The University response to Network Integration was much lower (36%) than the market average. WHY? We would speculate that this low University response probably results from the mature state of their own networks and/or from the relatively higher networking skill sets found in this environment. Training is in second place, with 59% of mentions, and here the Universities lead, with 74% indicating that they would purchase this genre of service. In the area of Business Process Design, the Universities are almost three times as likely to purchase this genre of service than all other institutions.

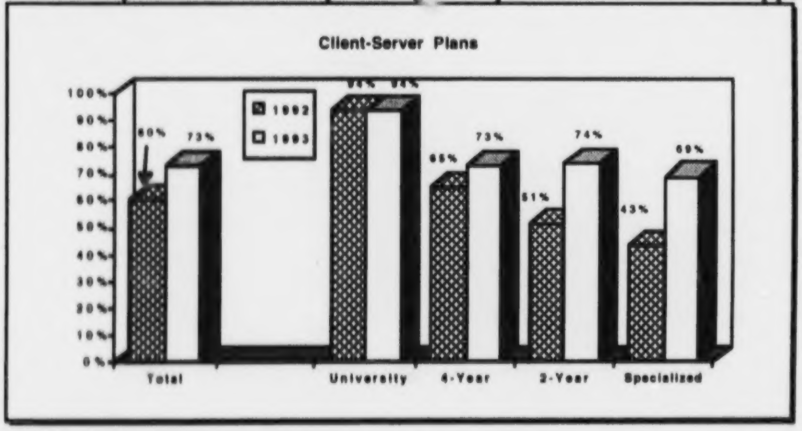
Which Issues Will Have the Most Strategic Impact?



When asked to rate the importance -- from least to most -- of several issues on their institution's strategic plans, the highest rated response was for Multimedia, with 55% rating it either "Very Important or Most Important." It is followed closely by Move To Client-Server (46%). While Outsourcing anchors the bottom of this list, in a related question, we found that 63% of all institutions currently outsource some aspect of their computing operations. Today, more than half of all institutions outsource hardware and software maintenance, and the forecast is for continued growth in these areas as well as in network infrastructure and maintenance.

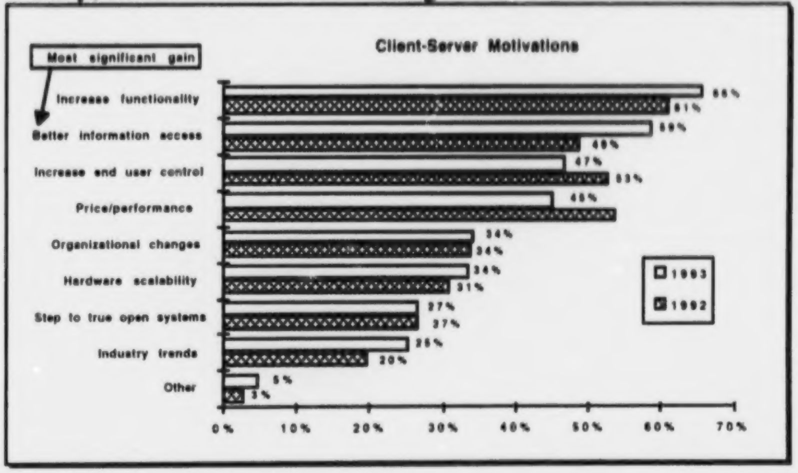
Client-Server Plans

How many institutions are planning to implement client-server applications? MOST! Overall, 73% -- a significant increase over last year's response of 60% -- indicated that they were planning a move to client-server. And, in contrast to last year, when there were significant differences across the Carnegie classes, this year only the Universities, with 94% planning for client-server, stand significantly apart from the crowd. The remaining institutions cluster reasonable close to the market average.



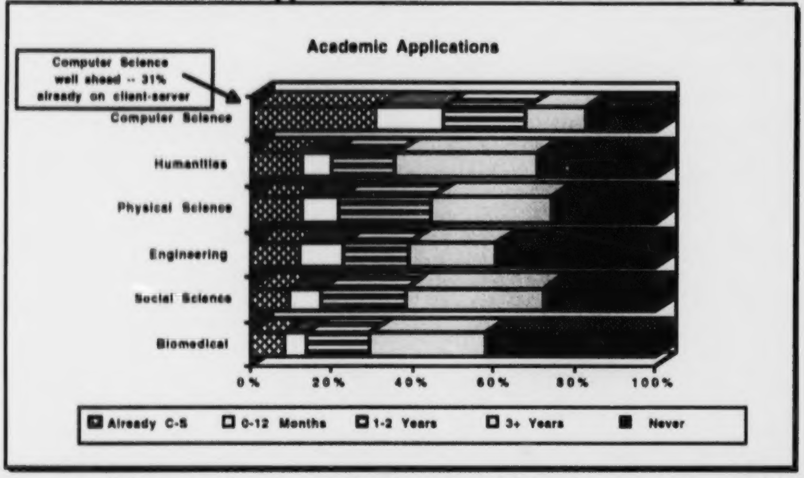
Client-Server Motivations

Clearly, most institutions are looking to move to client-server. The question is WHY? We asked respondents to tell us about their motivations. Increased Functionality tops the list, with over 60% of mentions, for two years in a row. The most significant gain is in Better Information Access. In both cases, the University responses were higher (82% and 78% respectively) than the other Carnegie classes, where the responses tended to cluster closer to the market average. In contrast, Price/Performance fell from #2 in 1992 to #4 in 1993, and the pattern was consistent across the Carnegie classes. WHY? We would speculate that as schools implement client-server applications, they come to understand that client-server may *not* be less expensive -- but instead a better -- way to define an IT architecture that meets institutional needs.



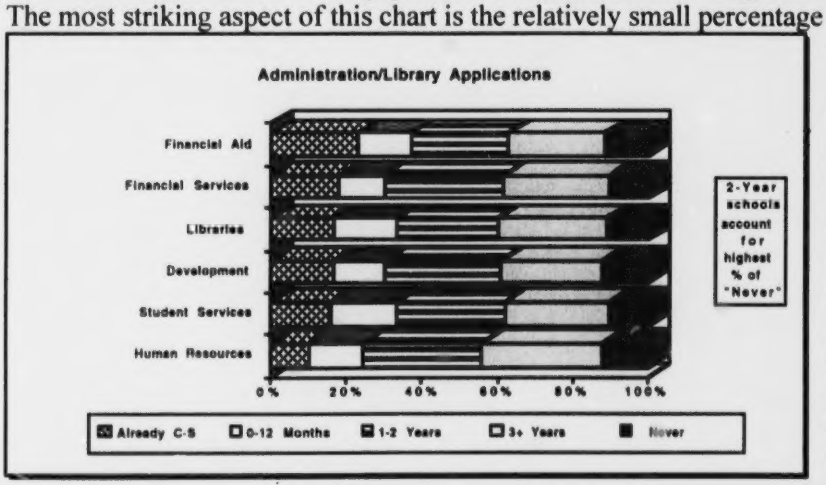
Client-Server Timing

When will Academic applications move to client-server? The good news is that a surprising number are already there. Computer Science leads this group, with over 30% of its applications currently running in a client-server environment. By adding those expected to migrate within the next 12 months, we can project that approximately 50% of Computer Science applications will be client-server based next year. What is the prognosis for the remaining Academic applications? Adding the percentage which are already client-server, to those expected in the next twelve months, we see a range of migration patterns from low of 14% in Biomedical and the Humanities to a high in the low 20's in Engineering and Physical Science. Social Science, with an expected 17%, is expected to straddle the middle of the range.



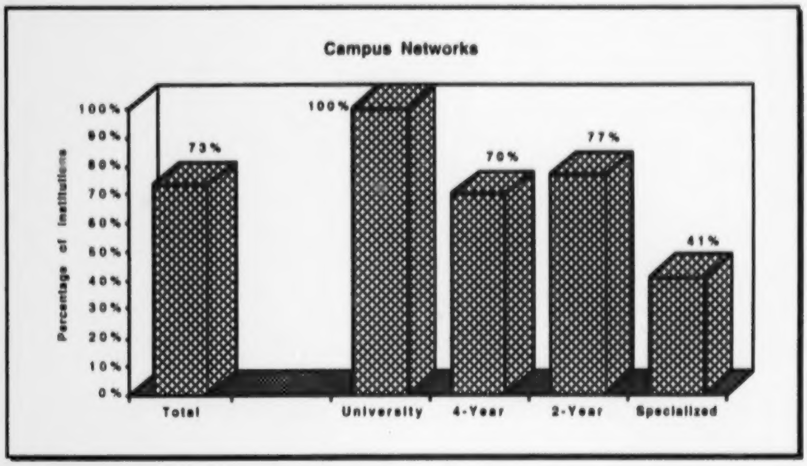
Administrative Applications Moving to Client-Server

The most striking aspect of this chart is the relatively small percentage of difference among the various Administration and Library applications. They all appear to share a relatively similar status for both the current and projected rate of client-server adoption. Financial Aid, with 23% currently client-server, is only *slightly* ahead of all others. Looking ahead to the 12-month projections for other Administration and Library applications, we should expect to see application migration to client-server ranging from a low of 25% for Human Resources to a high of 33% for Student Services and Libraries. The Universities, with their large and complex Administration applications, are expected to lag behind all others.



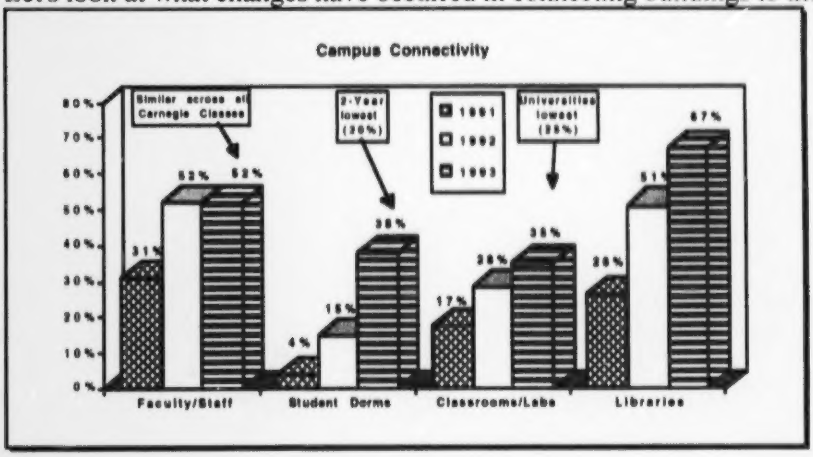
THE Issue Is Access

Ubiquitous access to information -- be it administrative, student or scholarly research -- is the pivot about which the Higher Education "wheel of fortune" turns. With access comes the ability to: empower work groups with the information they need to make decisions, enable administrators and advisors with real-time information to resolve student issues, encourage collaborative research, and facilitate the learning process through student and faculty access to e-mail, bulletin boards and external databases. How are institutions responding to this never-ending demand for more access? We believe that there are four dimensions with regard to access: (1) the presence of a campus network; (2) the provision of a wide array of network services; (3) the relative accessibility of those services by network users; and (4) the extension of network access to all members of the community. Our data addresses the first three and we begin by looking at the current status of campus networks. We have estimated that 73% percent of all institutions have a campus network.



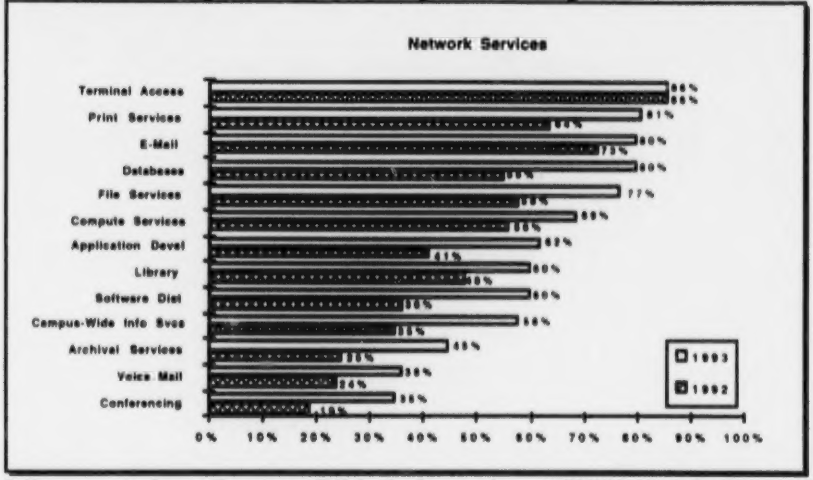
What's Connected?

Let's look at what changes have occurred in connecting buildings to the backbone. We have seen an increase -- over the last three years -- for all space categories. Faculty/Staff offices top the list, with 52% of those spaces connected, and this proportion is relatively similar across all Carnegie classes. In Libraries, where 67% are connected overall, the Universities lead. Likewise, they also lead in dorm connectivity, but they lag behind all other institutions in Classroom/Lab connectivity.



Network Services

The name of the game in networking is delivering services to end-users. What services are schools providing to network users? Overall, we found growth in the provision of most services over the last year. Terminal access and Electronic Mail lead the list, with well over 8% mentioning these services. There are also several other community-wide services which grew last year. Approximately 60% provide access to the Library catalog, Campus-Wide Information (CWIS) and Software Distribution. The emerging applications -- like Voice Mail and Electronic Conferencing -- are also growing, but currently have the lowest rates of provision.



Further, it appears that when institutions offer network services, they are accessible to a majority of network users. Over seventy percent of network users have access to: E-Mail; Voice Mail; Compute Services; Terminal Access and Print Services From the perspectives of both the provision of services as well as from the proportion of network users with access to them, institutions have made great strides. Perhaps the next challenge facing all institutions is to provide network access to all members of their communities... Education is, after all, about ACCESS!

Conclusion

Technology is not an end unto itself, but rather a means to an end... and hopefully a means to a "new end." It provides a scaffolding to launch new ways of thinking about -- and implementing -- the education mission. In order to deal with the enormous changes in the economic, political, cultural and competitive environment of the 90's, an increasing number of institutions are re-engineering their organizations from the perspectives of management, teaching and research. In turn, this organizational transformation is driving Information Technology priorities.

And, the priority should begin with ACCESS. It is THE major issue. In fact, broad-based access to all forms of information may be the delineating factor in predicting which institutions will thrive and which will teeter on the edge of survival. In the information age, survival of the fittest may be redefined as survival of the "most connected."

Desktop Information Delivery for Effective Administration Client/Server Solutions

Gary M. Hammon, Manager University Information Resources

University of Massachusetts
The President's Office
Amherst, Massachusetts

ABSTRACT

Current and proposed changes in the financing and management of higher education are fueling an ever increasing demand for information at the desktop of administrators and executives.

Client/server solutions can deliver critical information while enabling the IT organization and administrative customers to maximize computer resources, and more importantly, limited human resources.

This paper provides guidelines and strategies for implementing client/server solutions for management reporting, decision support, and executive information systems. A practical framework for using client/server architecture will be discussed with examples of applications implemented in a complex multi-vendor environment at the University of Massachusetts.

Introduction

CAUSE 93 comes at a time when the need for information to manage and support the mission of our institutions has never been greater. Regulators, legislatures and the public expect clear information describing the effective operation and value provided by each institution. This was highlighted in the June 1993 SHEEO/NCES Network Conference (State Higher Education Executive Officers/National Center for Education Statistics) which discussed important changes at NCES. "These changes have resulted from several causes including technological developments, the emergence of additional federal and state data needs and reporting requirements, and organizational changes within NCES and other agencies."(SHEEO/NCES,1993)

As Information Technology continues to evolve, there are more options available to provide the management information that is in such high demand. This paper provides guidelines and strategies for implementing one such option, client/server applications, as a solution for management reporting, decision support and executive information systems. The paper concentrates on the business issues and success factors rather than any specific technical designs.

The Challenge

Much has been written of the rate of change in information technology, especially in view of the ability of systems to offer meaningful information for management of the enterprises. Without regard to the debate of evolutionary change versus revolutionary change, we need to take stock of the situation at our institutions and identify rapid deployment strategies to deliver the critical information needed today, (or more likely in our customers eyes - yesterday). While our strategies should also have an eye to the future we must avoid the inclination to design the perfect system - time is of the essence.

The Changes

The 1980's saw extensive growth in our legacy systems and data stores. Disk storage costs decreased, new programmer productivity tools and methodologies became more effective, and we began to see data as a resource to be managed! Of course like all valuable resources, we saw more as better, so the data stockpiling combined with cheaper storage combined to create massive data stores.

PC's came on the scene and continued to explode through barrier after barrier: faster, cheaper, and "ubiquitous". The talk was of the number of PC's per worker, and the forecasts of deployment to the desk of every knowledge worker appear to be very accurate. The 80's were also a time of limited use of graphics in the everyday operation. Graphics were the things of special occasions - annual reports, etc.

Current Management Information Environment

Now in the 90's we are often drowning in data. How do we turn this wealth of data into information? What is the cost of this transformation? Bill Inmon, regarded by many as the father of the data warehouse, has noted that the unresolved issue is the transformation of distributed data into usable information regardless of the platform.(Knight, 1991) The business areas at our institutions have acquired extensive technical capability. The staff are increasingly computer knowledgeable, especially using desktop computers. Many

Information Systems organizations have established the policy that the business areas are responsible for producing their own reports, a step which the University of Massachusetts President's Office took in 1992.

Regulators are also stocking their databases and expanding their reporting capabilities. Explaining findings derived from our "exported" data taxes our institutions ability to respond rapidly with in depth and comprehensive reporting and analysis. According to the National Center for Education Statistics, "Various legislative requirements are also leading to the development of new data sets for higher education.[such as] the 1992 Higher Education Re authorization Act, as well as the Student Right to Know Act of 1990 which give new importance to the need for accurate and comprehensive information." (SHEEO/NCES,1993)

Business graphics have become essential to credible communication, again, the result of years of desktop computer deployment, friendlier packages, more skilled users, and everyday use in newspapers, newscasts, business weeklies and even political debate.

Our current environment is one of demand for rapid information delivery. Increasing experience with the responsiveness of desktop computers has led the business user to view timely delivery of information as an entitlement that should be within the abilities of their information suppliers.

If we are to meet the information needs of our administrators, we must act now, leveraging our resources to deliver more capabilities in the short term, while building toward a service environment that keeps pace with the demand for management information.

Leveraging Strengths

At the University of Massachusetts we began by assessing our core competencies as they related to delivering information access and reporting in a client/server approach. That is, an approach that provides requested data from a designated source computer (server) to desktop workstations (client) applications. In our environment this was assumed to be a VAX server accessed by desktop computer application software. The areas of strength that we identified as essential to success were infrastructure, data, business area readiness, and management support.

Our infrastructure strengths were in the area of networking, a critical factor for our dispersed environment of six sites throughout Massachusetts. In 1993 we implemented a wide area network using a T1 ring. The networking services group has extensive experience in multiple protocols on the wide area network as well as experience in local area networks consisting of both Apple and IBM compatible desktop computers. Experience in configuring workstations, supporting desktop computer software, and administering a local area network for several years were also recognized strengths.

The university had made an effort to improve the management of the data resource through the efforts of a university-wide Information Systems Task Force, which adopted a resolution that data administration should be a priority. As a result, a University Data Administrator was named in 1992. The official focus on data as a key resource led to improvements in the data dictionaries for the primary systems administered by the President's office. Particular attention was devoted to the Human Resource Management Information System(HRMIS), which provides all payroll and on-line human resource processing for the university. In addition to supporting this processing with operational reports, HRMIS also provides detailed data to the Commonwealth of Massachusetts.

In 1992, University Information Systems senior management decided to form the University Information Resources organization. This organization was charged with promoting University-wide data administration and implementing a data warehouse to improve management reporting capabilities. The primary business area supported by the University Information Resources is the President's Office. Secondary support is provided to the campus administration through the Vice Chancellors of Administration and Finance, Institutional Research Office, Budget Directors and Human Resource Administrators. Notably, these constituencies were making extensive use of desktop computing and, in many cases, batch ad hoc report generators accessing legacy system extracts.

While there was a history of commitment to management reporting and the development of a data warehouse, the work prior to December 1992 was limited to research and fact finding. The activities from December 1992 through the Fall of 1993 included implementation of the data warehouse, client/server reporting, ad-hoc reporting and an Executive Information System. The strategies and lessons learned may offer ideas and examples of how success can be achieved in developing and implementing client/server reporting to promote effective administration.

A Solution in the Real World

Given the pressing business demands, it was clear that a protracted development process was not a viable approach. While there was preliminary support for the concepts of data warehouse, data administration and management reporting, it was clear that continued funding would require visible tangible results.

Certainly the core of this client/server project was the business need for management information. The strategy employed was to identify a single area of business need, and focus the project on implementing a generic solution. It was important that solving the particular business problem require us to address the essential components of client/server access to a data warehouse.

One of the early advantages realized through our focused approach was positive movement toward a clear goal. I contrast this to a protracted planning process that might require an early consensus that is extremely time consuming. An example of a task that could delay progress is deciding the full range of data to be included in a data warehouse. The answer to political issues such as this could easily require months, if not years, of meetings with representatives from many functional areas. Our strategy is to develop the data warehouse in stages, consistent with an evolving University Data Model.

One problem area that came to light in early meetings with key business managers was that of human resource information. Some standardized monthly reporting had been developed to provide key measures and indicators for comparison across the five campuses. However, the reports only provided information for the current month. Over a period of three years, a spreadsheet application grew around this monthly data. The application, which was developed in the President's Office, grew to a set of eight reports that accessed an Excel database to calculate comparisons to prior periods, average salaries, and permit trend graphing. The data entry from the standard production report, and the manipulation of massive spreadsheets made this a high cost application which was identified in several discussions as needing a "better approach".

Since the business area staff had clearly established Excel as their standard, we decided that the initial client/server implementation should be able to deliver data to Excel spreadsheets with minimal user effort.

The decision to implement a client/server solution was made basically from the requirement to deliver a solution that offered easy access to critical data, and recognized the need to provide for more extensive reporting capabilities. Some of the clearly identified reporting needs were in the area of more detailed Human Resource data as well as financial and academic data that knowledgeable business staff could combine as needed without specialized technical skills.

The need to provide this type of cross functional reporting clearly indicated a relational database would be required. While the majority of our legacy systems reside on an IBM mainframe, the IBM environment does not have a relational database environment. However, our VAX environment had Rdb as an installed supported relational product. During some of the early research into improved management reporting strategies, Digital Equipment Corporation had provided extensive information describing their implementation and use of Rdb for massive data warehouses. As previously noted our networking capabilities and support were very good, especially in the options available through our VAX environment. We decided that the warehouse would use the exact data that was reported in the standardized monthly reporting. This would provide more detail than was available in the spreadsheet database, and would allow us to start the warehouse with over 30 months of history.

The general information needs identified were: ad hoc access to produce summary reports and comparisons, the need to provide executive information, with perhaps some direct hands-on executive access, and an easy way to move data directly into Excel spreadsheets for manipulation and graphing.

Our early research had identified that Data Access Language(DAL) was a reliable widely supported facility to use on the VAX to enable client machines to access data stores on a VAX platform. Testing indicated that indeed Excel and DAL worked quite well together, so the combination was identified as the first implementation of client/server reporting from the data warehouse.

The decision to utilize Data Access Language(DAL) precipitated a number of other decisions. One notable area of some uncertainty was software for ad-hoc reporting. Years of experience in non-workstation based access tools indicated that better service to our customers would require a client/server solution. Once DAL was identified as a standard client/server technology, several DAL compliant products were identified that would provide highly flexible ad-hoc reporting.

In summary this project answered a critical data need expressed by some of the top administrators in the institution, by providing easy access to well defined data using familiar tools. This has proven to be as successful an implementation as this summary would suggest it should be. A variety of other factors contributed to the success of the project. The remainder of this paper describes some specific lessons learned and provides guidelines for other key success factors.

Lessons Learned

1. Once the decision-making process began, a clear framework emerged, with of course certain limitations . However, initial decisions such as selecting the database management system, focused the thinking of the team on implementation tasks and time frames. It also become much easier to identify activities that were not essential to the initial implementation.

2. Client/server need not be viewed as a radical change of direction, but rather an extension of other initiatives. We had clearly decided to implement a data warehouse, but had not established that the initial access would be client/server.

3. It is not practical to become expert at everything that is listed in the technical critical success factors for client/server. You may need to develop alternatives such as an interim solution, or a partnership with an expert.

One of the key components of our data warehouse is an encyclopedia. Our design called for the encyclopedia to be integrated into the desktop environment, such that business rules and definitions could be viewed, and queries could be generated via navigating the encyclopedia; Aetna Insurance has developed a facility that is highly successful. However, this functionality could have added as much as 6 months to the implementation, so we developed some standard queries that could be invoked to provide the core information. Since our Rdb expertise was lacking, we partnered with DEC who provided some long term design assistance, as well as some rapid delivery of Rdb services.

4. The specialized knowledge, the need to have innovative approaches, the need to build ongoing support in the IS group requires a cross discipline team. We included the University Data Administrator, a networking and architecture specialist, and an end user computing expert as part of the core team. This worked very well in raising the concerns necessary, and in building commitment in these essential areas.

5. Partnerships with user constituencies are invaluable. We plugged into some key university-wide groups that would be stakeholders in the data warehouse, such as Institutional Research and Human Resource Administrators. The number one concern of these groups was ensuring proper definition and understanding of the data that would comprise the data warehouse. Groups such as these offer help in definition work, as well as provide essential feedback. One way of gaining support for data warehouse, or similar activities that make data accessible to management, is to make the implementation a "win" for these constituencies. Whatever capabilities we provide for the President's Office, we also provide to the campus. Although the campus may already have the data, the access via a new client/server tool may be welcomed. For example campus access to President's office EIS reports and graphs, provides these facilities without any development cost, limited of course to the data that the campus is authorized to access.

6. You do need to pay some attention to the "open systems" approach. Open solutions can provide flexibility and lower implementation costs. Open architecture, open systems is really driven by what your environment is, and what you expect it to become. Why wouldn't you strive to have an open solution? Do I really want to develop one application for Macintosh and another for DOS platforms? These are some of the issues which client/server forced us to answer. We have decided to require portable applications. This is working out quite well using open solutions such as Excel's database interface, and the HOLOS Executive Information System. You may also find that by starting into client/server, you will actually act as a catalyst for the organization to set a direction toward open architecture.

Choosing Initial Applications - The Business Case

The popular phrasing applied to newer approaches seems to be, *"it's more of an art than a science"*. However, there are some basic approaches that can make choosing an initial client/server application more science than art. No doubt as your reputation, time and money are going to be invested in the first client/server application, it would be nice to find a "sure thing". The easiest aspects to address is what types of applications should be avoided.

If client/server is new to your organization, there are a number of areas that must combine to make the effort successful. Although client/server has some similarities to what you might have done in the past, it is deceptively different. It is more than installing a new mainframe reporting package, it is more than installing a new PC package, or a new network access, and yet it often includes all of these activities. For this reason, it is recommended that the initial application should not be: mission critical, date driven, or a large application. A somewhat more conservative approach is to "concentrate on a class of applications that do not require real-time data access between the microcomputer and the mainframe because these applications are more easily manageable, fit better with the current skill level of most organizations, and can be delivered more quickly." (Capraro, 1993)

Aside from the inherent risks in these projects, cost can be a significant issue. Frank Dodge, co-founder of McCormack & Dodge Corporation, in responding to a question on cost savings realized with client/server solutions, indicated that "it's a mistake for companies to believe that overall costs they allocate to a particular set of applications will dramatically decrease within the first eighteen months or so." (Dodge, 1993)

While the technical components are clearly important, the best technical development of a low value application will not produce a successful result. One of the best approaches to finding a high value application is to be engaged with the business areas you support. Involvement in the business issues and understanding the demands being made of the business areas, provide a basis for answering the central question "*Where can I provide a new information service that gives the business area a boost in productivity or information access?* "

Why consider productivity first? The stories are legion about the reams of paper that are printed from legacy systems, distributed to the business areas, and then used as a source for a few key data entered into a desktop computer. The desktop computer can then be used to produce the report that has real value to management. I would suggest that this is perhaps one of the top candidates for an initial client/server application.

Looking at some key success factors shows why a focus on this type of data re-entry situation can offer a high probability of a successful client/server solution.

Business Need - With resources generally limited, the reporting topic is likely to have some importance if it is being maintained in spite of the cost and delay of data entry. The topic area offers opportunity to improve quality, timeliness and the scope of the data available for analysis and accuracy.

Well Defined Data - Whenever business area access is provided to a data source, the data should be clearly defined. Client/server can accentuate the need since the particular solution may require distributing the data to local servers. Data that is reflected in standard reports is often defined in the "business rule" or logic that produced the report. In addition, data that has been available for a period of time in a standard report usually has an accepted definition. The availability of these types of de-facto definitions enables the client/server project to proceed without needing to invest immediately in potentially lengthy data definition sub-projects.

Easily Accessible Data - The fact that report programs exist provides an easy opportunity to extract, at a minimum, the same data that is included in the standard reports. Data distribution and access policies can be less of an impediment if the initial client/server application is simply deploying new capabilities to areas that already have paper access to the data.

Historical Data - While typical operational system reports focus on the management of the business function, operational reports do not provide the type of trend information needed for decision support and executive management. Often, these management reports require a business area to key enter data to create local private "databases". A client/server solution can address this need, by providing a much easier means of drawing on the data, especially if the solution is to convert the data to a relational data store.

There are two areas of human factors to consider in choosing the initial application - the business partnership, and the staff expertise.

Business Partnership - If management in the targeted business area is an ally in the project, then the likelihood of success is increased. While building on past successes is a definite advantage, strong allies can be established by developing a shared view of the benefits. One way of enhancing the benefits is to have as little disruptive impact on the business area as possible. The single most important area to address in minimizing the impact is to avoid solutions that require major changes to the business area, such as conversion from PC to Macintosh, or from one spreadsheet package to another.

Staff Expertise - Client/server is an evolutionary step in information service delivery, particularly for reporting applications. This means that there are similarities to many of the services already being provided. Staff who have been heavily involved in building databases, supporting ad hoc data access via 4 GL tools, supporting desktop tool installation, and system design are key resources for a client/server application.

In addition, skills are required in networking and desktop network access, which may necessitate expanding the team beyond the system development group. At the University of Massachusetts, the data warehouse team, responsible for implementing client/server reporting applications included key infrastructure staff. These infrastructure experts addressed the networking, desktop connectivity and many open architecture design issues.

Partnerships with vendors - Vendors can bring very specific expertise to the project, which allows the team to focus on strengths, rather than trailblazing in new technical capabilities. One example of vendor support which was highly effective in the University of Massachusetts' client/server implementation was a one day project review with a Digital Equipment Corporation Team. The DEC team included experts in relational database design, wide area networking and desktop connectivity. The review provided some expert advice, and an extra measure of confidence in our project plan. We later contracted with DEC to develop an EIS client/server application using the HOLOS software developed by Holistic Systems Corporation. While the University of Massachusetts team was responsible for all other components of the EIS project, it was extremely helpful to use DEC's expertise to develop our first HOLOS-based application.

Cost Savings and Justification

The debate of whether or not client/server and downsizing save money will probably continue for some time into the future. However, in focusing on reporting applications, we can rely on some clear business issues to determine if client/server is a cost-effective solution for a particular information need. We can consider some of the common applications and issues to illustrate how cost and value might be presented to support a decision to implement a client/server solution.

A key question to answer is "how is critical reporting being addressed?" If data is being key-entered to desktop computers to create reporting databases, there are some associated costs. In addition to the data entry cost, there are some quality control costs in checking for keying errors. If the keying errors are not detected, what is the cost of decisions based on incorrect data? While this cost cannot be quantified as easily as key entry time, raising the question may be a powerful factor.

Data entry report applications can be problematic due to the inherent delay in producing the report, and the potential loss of some important information that was deemed "too much" data to be keyed. These situations lead to decisions being made without the benefit of a key resource, the institutions' data. There is also a high potential for duplicate effort since many recipients of a valuable standard report may use the report for similar purposes such as tracking the same trends via their desktop computers.

The nature of management reporting applications often implies use of central data stores with distributed access, a standard model. When this model is not used there is increased risk not only of duplicate effort, but of arriving at different answers to the same question, depending on the design, quality, and currency of the redundant, locally created data sources. Even given consistency, if complex rules are needed to evaluate or manipulate the data, these rules need to be replicated for each locally created data source, further increasing the risk of erroneous results.

A key characteristic of client/server applications is the familiar and comfortable desktop access they provide. Training business area staff to access data from graphical interfaces and desktop query tools with context-sensitive help is much simpler and faster than delivering training and support for 4 GL languages. The cost savings for the business area is in both the reduced training time and the more effective use of the reporting capability; the difficulty and time spent making 4 GL's produce desired reports and graphs is dramatically reduced in client/server approaches.

While there are a variety of ways that cost savings could be approached, improved information quality and effective use of scarce resources, particularly in the business areas, can be a powerful illustration of the benefits that can be derived from a client/server approach to management reporting.

Implementation Strategies

Creating a Focus

Once you have decided to proceed with a client/server solution to a key information reporting need, it is essential to create a focus that will assure success.

Although the justification for the project may be that a number of business areas will benefit, the initial implementation should be limited to a small number of customers in the same location. Allowing for a few problems is realistic and quite manageable if you have targeted the customers who will succeed with you. Undertaking the initial client/server implementation with a customer who does not share your belief in the project, is inviting intolerance and negative publicity that can brand a project or technology as a failure.

One way to maintain the targeted area's support for you and the project, even if you are experiencing some delays or problems, is to be highly visible in the customer's area. Frequent demonstrations of prototypes, validation of workstations, networking tests, etc. all show that the project has commitment, and invite a supportive reaction from the customer, even if the project is off schedule. Again, it should be emphasized that the potential for problems is one of the reasons that the initial client/ server application should not be "mission critical".

Client/server applications are well suited to prototyping and therefore promote satisfied customers, provided that the developers pay close attention to customer reaction to issues such as ease of use and features the customer deems critical.

Marketing

The initial client/server application represents an important event in both the business and Information Systems organization. A successful implementation will require support from many individuals in these organizations. Since people can not support what they don't know about, it is important to keep these constituencies informed.

Using a variety of communication opportunities, as often as these opportunities are available, will help to ensure that everyone is aware of your client/server project. Updates in newsletters, staff meetings, departmental meetings, and informal updates are all effective in ensuring that the project is visible, and a recognized priority.

An important aspect of the communication, particularly for the business areas, is to set realistic expectations, and continue to reinforce these expectations regularly. This is especially important if the implementation will occur at only one of several planned sites well in advance of the implementation at the other planned sites. In this situation, and as a general means of sharing information, demonstrations of the client/server application are a powerful communication tool. Since an amount of skepticism is often attached to new initiatives, demonstrations serve to dispel the notion that the whole effort may be vapor-ware. Demonstrations also help to clarify issues and surface key concerns before the project is delivered. For this reason, demonstrations within the Information Systems area may be conducted very early in the project and may show only product features, since application and development may be in very early stages.

Once the project has reached final testing stages, demonstrations take on a different role. At this point they can serve as a "wake up call" to everyone who will be involved in the implementation. This can help to create a push to wrap up any outstanding tasks. The demonstrations of production-ready systems also create enthusiasm in the business areas, which is important to generating more interest in future applications.

Infrastructure

The network plays a key role in client/server applications since the client workstations must be able to communicate precisely with servers, whether on local area networks, wide area networks, or a combination. For this reason, the infrastructure support is more critical to success than ever before.

New areas of support that should be considered with networking include workstation configuration and workstation product installation, particularly for products that support the network communication. Whenever possible, the infrastructure support areas should be offered the opportunity to participate in product selection.

Involving these areas at the inception of the project and ensuring a continuous update process will pay dividends in a client/server project.

Summary

While client/server development can appear daunting at first, there are strategies that can be used to introduce this technology with limited risk and expense. As with any new technology, success should be measured by the value that the application of the technology brings to the institution rather than the inherent potential of the technology. This paper has attempted to provide some insights into one approach that was highly successful in delivering business value to the University of Massachusetts.

References

Capraro, Anthony J. "So You Want To Downsize Your Applications?" Information Strategy, Spring 1993 pg. 54

Dodge, Frank "Client/Server Technology: Looking Beyond the Mainframe." Global Business Review, Digital Equipment Corporation, 1993 pg. 25.

Knight, Robert. "DBMS - Facing '90s Obstacles." Software Magazine, v .11, December 1991, pps. 27-28.

SHEEO/NCES Communication. "Network News." v.12, no. 3, September, 1993, pps. 1-9.

From Server to Desktop

Capital and Institutional Planning for Client/Server Technology

Richard Monty Mullig
Keith W. Frey
University of Chicago
Biological Sciences Division and Pritzker School of Medicine
Management Information Systems

I. Project Goals

In 1988 we initiated a project to provide enhanced decision and strategic planning support to senior management in our Division. The impetus behind this project was senior management's desire to integrate information from disparate University systems in complex reports while avoiding the high costs associated with mainframe computing. To fulfill this need, we developed an integrated database encompassing many domains of administrative data and a suite of reports. As this project proceeded the potential power of combining this database with emerging new technologies in networking and graphical user interfaces became apparent and the original undertaking was expanded into a full-scale client/server development project. The range of services to be provided was enlarged to include not only decision support services but also applications to automate routine administrative tasks, ad hoc reporting for administrators, internal communications tools such as electronic mail, and higher quality desktop computer tools and services. All of these services have been delivered during the past five years using client/server technology.

In this paper we will review the major planning issues we faced during our project and consider several important factors in implementing client/server technology in general. First, we will discuss how and why the scope of the original project was expanded beyond its original boundaries. Then we will consider each major phase of the project in turn, presenting the planning and implementation issues that we faced during each one and the critical factors for success. While we will touch on staffing issues throughout the paper, in Section VII we will describe in detail the challenges we faced in organizing our technical team and the processes we adopted to overcome them. Finally, we will conclude with a summary of the overall costs of our project, with particular emphasis on the distribution of those costs and the implications of the costs and their distribution on institutional capital planning.

II. Early Organizational Issues

At the outset of our project, senior managers in central Divisional administration were the exclusive consumers of our services and little thought was given to extending these services into the individual departments that make up the Division. Indeed, central computing staff were reluctant to foray into the departments because we believed that departmental support could be best provided locally. Two factors led to a change in this attitude and a broadening of our client/server initiative to include the entire Division.

The first of these changes was a shift in management style that accompanied a change in senior management personnel. Whereas in the past the central administration had formed a rather closed circle and each department operated more or less autonomously, the new management of the Division fostered a more open style and encouraged close interactions between the departments. One important aspect of this approach was the appointment of a focus group on information systems. This group was charged with identifying opportunities for departments to collaborate in developing solutions to administrative computing problems. The formation of this group rapidly led to much closer working relationships between departmental administrators, who also began to look to central administration for leadership and assistance, especially technical assistance. This new atmosphere of cooperation and shared interests provided an essential foundation on which future organization-wide systems efforts were based and provided these efforts with both an engaged group of users to guide the projects and a source of "early adopters" for new systems.[1]

The second change that contributed to the acceptance of our client/server initiative was the successful implementation of a central, integrated database and the success of the networking technology deployed in central administration. While these two technologies were originally targeted only to support decision making and strategic planning centrally, the utility of these technologies to the departments was apparent, and their early success gave the technology and the technical staff of the Division credibility. Coupled with the new management style in which departmental administrators cooperated to solve problems and shared successful tools, our more ambitious plans for an organization-wide client/server initiative were put forward in a receptive environment that might otherwise have been hostile or disinterested.

III. Project Planning Issues

Given the modest initial scope of our project, we did not plan to deliver full-scale client/server access to our database and, although we anticipated the widespread use of networking in our Division, no strategic networking plan was put in place. Within a year, however, as we completed the most significant segments of our database, the need for networking services, became apparent and pressing. This need was driven at first by the demand for electronic mail and later by the need to deliver interactive access to our database to administrators throughout the Division. When we considered the requirements of our newly expanded project, four critical components emerged: a system architecture with three technical layers and a team of personnel to implement the project. The three technical layers were the data repositories, network infrastructure, and desktop computer environment. Since a significant portion of database repository had already been developed by the time the project was expanded, early work focused on developing the network infrastructure. Work thus proceeded in a bottom-up fashion, with servers and networking receiving attention before specific desktop applications. This approach served the project well, since we were able to deliver rapidly several new services to our users once the infrastructure began to take shape. One such service was electronic mail which, while not the ultimate goal of the project, proved extremely popular and generated tremendous support for our efforts in the organization.

Although the original goals of our project dictated that we develop the database and server components first, we believe that client/server projects would benefit from this phasing in general. While all three components of a client/server implementation could be developed in parallel, this approach is difficult for several reasons. For example, if the server and network infrastructure is not in place during client-side application development, programmers will be forced to rely entirely on the infrastructure specifications and may therefore lack an adequate application testing ground. Also, in our project (and probably many client/server projects), staffing was limited and it simply was not feasible to have three distinct development efforts underway concurrently. Where efficiency or other constraints force an organization to address its development serially, the most stable and scalable technologies should be addressed first followed by the more volatile and vertical technologies. Our experience shows that server and network infrastructures are both relatively stable and have had, at least over the last six years, far longer lifetimes than desktop computer and user interface technologies. Thus, in our project and, we believe in other complex long range projects, success is best fostered when effort in early stages is concentrated on deploying stable technologies and is gradually shifted into the more volatile technologies as the infrastructure falls into place. Our expectations that server and network technology would remain stable while offering rapidly increasing levels of performance proved accurate, and today, nearly six years after the project began, our server and network technologies, as well as our original data model, continue to provide a solid foundation on which new client applications can be readily developed.

---

[1] The decision to undertake strategic information technology initiatives is often difficult to justify quantitatively and this lack of quantitative evidence often inhibits organizations from undertaking strategic projects that may be critical to their success. For a discussion of how these decisions can be made sensibly in the absence of hard, quantitative support, and the importance of both strong leadership and cooperative risk-sharing in strategic technology deployments, see Clemons, Eric C., "Evaluation of Strategic Investments in Information Technology," *Communications of the ACM*, January 1991.

IV. Servers and the Integrated Data Repository

Relational database technology was chosen to implement the integrated database because of its powerful data modelling capabilities and its expressive data manipulation and definition language, SQL. We also desired a rich programming environment while minimizing the number of platforms our technical staff would be expected master. This platform also had to be capable of supplying a full range of network services over our chosen network protocol, TCP/IP. Finally, we had a strong preference for working in C, because of its support for structured programming methods as well as its flexibility and power. In many ways, these criteria amounted to language decisions which clearly limited the field of potential platforms and environments and had a great influence on the design methods we would later employ.

In our case, we chose Ingres (and later Sybase) for our DBMS, and Sun servers running Unix as our platform for database and other server services. Although our server technology has remained relatively stable and consistent over the course of our project, the potential offered by "open systems" to replace a component supplied by one vendor with another supplied by a different vendor was also very attractive to us. And, in fact, we did make one major change to our server technology during the project. After originally developing our database using Ingres we chose to migrate to Sybase once our full-scale client/server development began and we re-examined our DBMS requirements. While this decision allowed us to implement our client/server systems more easily and greatly enhanced the performance of our database, it did entail significant migration costs. Even though the programs we developed during the 18 months we used Ingres were written almost exclusively in C and SQL, migrating these programs to Sybase required a substantial effort, equivalent to about 30% of the resources expended during the original Ingres development phase. In addition, because staff had moved on to other projects and had to be reassigned to complete the Sybase migration it took over one year to complete this transition. Our experience contrasts with the often emphasized promise of ready portability between "open" systems (such as C, SQL, and Unix) and has tempered our willingness to consider changing vendors for our other server technologies.

Our selection of Sun Microsystems' hardware and operating system for our hardware platform made available servers of various capacities, ranging in size from small desktop units to large rack-mounted machines. This flexibility and the successful implementation of these systems in central Divisional offices, along with the University's disposition towards departmental autonomy, encouraged individual departments to install servers and local networks in their administrative offices. Central computing staff did not discourage these installations because departments who installed servers made a visible commitment to the project which carried much political significance. Consequently, the demand for Unix system administration and end-user support burgeoned as servers and networks proliferated throughout the Division. Many departments hired local technical staff to provide these services, but because the skills necessary for Unix system administration and end-user support are very different, and the demand for end-user support was often overwhelming, the quality of local system administration varied greatly from department to department and was sometimes very poor. In response to this situation, we have begun to consolidate both system administration and servers within the Division, allowing departmental technical staff to concentrate on end-user support while achieving economies of scale in the one area where they appear most realistic in a client/server environment, namely in the purchasing and administration of file and database servers. This need for specialization and its impact on our staff organization recurred in many areas of our project and we will explore it in detail below.

V. Network Infrastructure

We elected to adopt the TCP/IP protocol suite running on Ethernet as our enterprise-wide networking strategy. This strategy met our need for a network infrastructure that could support campus-wide connectivity[2] and was well supported by the relational databases of the time.

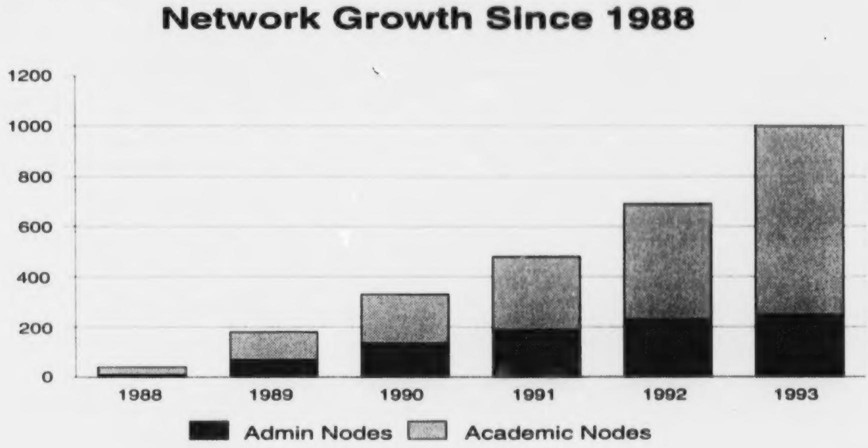
The network was implemented with two initial aims: to extend the backbone to cover more area, and to install the network in the early adopter departments that had also committed to our server strategy. Although the backbone was funded centrally, local area networks were funded by the users, either departmental offices or individual faculty. While this approach was adequate for the initial deployment, as the network began to grow very rapidly (spurred largely by the discovery of electronic mail) our ad hoc approach began to unravel. The technical staff responsible for extending the network was small and was also responsible for developing the database and server infrastructure. Their ability to meet the demand for new network installations was severely strained.

It is our belief that this funding mechanism for network expansion was the most significant factor in our failure to develop a strong central staff capable of properly planning and managing our network. Network planning and management is largely an overhead cost, and our funding mechanism did not provide a method for recovering these costs, which are quite significant, and users resisted our attempts to introduce service fees or other cost recovery methods. Without a ready, self-sustaining source of funding for the staffing and management tools needed to plan, design, and implement our network infrastructure, both the MIS organization and senior management were reluctant to channel resources into this effort. Even as our technical support organization matured, the lack of an adequate funding mechanism for network planning, coupled with the difficulty of explaining the benefits of such planning to senior management, inhibited us from ever fully establishing a network planning and support staff. In our experience, it is not fruitful to argue for such efforts using quantitative analysis, and other means of winning senior management's support must be found. Some useful techniques might include analogies to other services traditionally provided as overhead services or, perhaps even more effectively, using either scenario analyses and decision trees to illustrate the best strategy in the absence of hard quantitative evidence.[3]

---

[2] A TCP/IP Ethernet backbone was already in place on campus to provide inter-building connectivity and this allowed us to take advantage of the existing campus-wide network technology rather than reimplementing an entirely new network infrastructure in parallel.

The absence of a plan for installing and managing the network resulted in high fixed incremental costs for new network installations, while the shortage of staff left the network undocumented and unmapped. Despite these circumstances, the network has grown to reach 95% of all administrators in the Division and now serves nearly 1,000 faculty and staff (see Figure 2).



VI. Desktop Computers and Custom Applications

Delivering direct access to the database through intuitive and flexible tools to manipulate and analyze that data is the great promise of client/server applications. While the server provides a powerful, enabling service to client, the client is responsible for mediating all end-user interaction with the system and it is through the client that the ultimate functional benefit of client/server computing is achieved. Client applications for our database had characteristics of both decision support and transaction processing applications. Administrators needed to perform both ad hoc and routine, repetitive queries. At times users needed to incorporate the results of these queries into other tools, most frequently worksheet models. Finally, users also needed to be able to easily enter information into the database.

The critical role played by the client requires a more capable machine than a simple character-based PC and this fact was reinforced during our initial attempts to deliver client/server solutions for our existing stock of aging, DOS computers. Character-based DOS computers could deliver neither easy-to-use access to a wide variety of network resources simultaneously nor the development tools we needed to implement complex client/server applications. Instead, client-side services require intuitive graphical user interfaces to provide the user with simple, expressive, and powerful mechanisms for understanding and manipulating information. Large monitors are needed to allow the user to view information from multiple sources or through different applications simultaneously and to move readily from one application to another. Large virtual memory spaces are required to support both the graphical display and the simultaneous use of many different applications. Network support must be pervasive in the desktop computer; in the client/server environment, the desktop computer is less the private tool of the end-user and more the user's network access tool which transparently intertwines network resources with the applications running on the user's desktop. Finally, tools are needed for developing custom client applications that allow a small programming team to very rapidly develop and enhance applications that deliver the database to the user in a graphical interface.

---

[3] *Op.cit.*, Clemons, Eric C. In our University, network infrastructure services in the last year have been assumed by the University's telecommunications office. Nonetheless, planning and implementation of the network infrastructure remains ad hoc and service is inconsistent and often inadequate. In our view, this continued ad hoc and unplanned approach to network infrastructure can still be traced to a reluctance of middle managers to use non-quantitative analysis to support a request for proper funding to senior management and, for senior management, an over-emphasis on quantitative justification for strategic investment which is then communicated to their middle managers.

Our vision of the ideal client workstation stood in stark contrast to our installed base of DOS-based PCs and led us to re-evaluate our desktop computer strategy. In particular, we developed a coordinated capital replacement plan to invest in new desktop technology specifically capable of supporting a client/server environment. This approach to desktop computers was entirely new to our institution, where desktop computer purchasing had always been the province of the individual, rather than a part of an institutional capital strategy. With the aid of a committee with members drawn from departments throughout the Division and central University administration, we were able to develop a consensus for a desktop computer capital replacement plan centered around a minimum set of hardware requirements and a desktop Unix operating system featuring powerful object oriented development tools.

We have seen two major benefits from treating desktop computers as a capital, institutional investment. First, our ability to plan and design applications and services has been enhanced by a stable, well understood desktop computer strategy. No longer are we forced to stretch resources and expertise thinly, and inadequately, over several platforms all running different software packages. Instead, we can focus our support and development efforts on a single platform with a well defined growth path while, at the same time, avoiding the trap of the "least common denominator." Secondly, by treating desktop computers as an institutional investment rather than an operating and unit expense, we are able to raise the standards of capability in our desktop computers. While more expensive units are sometimes purchased for individuals under a capital plan than would be purchased on an incremental, ad hoc, operating budget, our desktop computer stock has become far more flexible than it otherwise would have become. All desktop computers in our organization are essentially interchangeable allowing them to be redistributed easily to meet our needs. Moreover, we believe that the marginal savings of purchasing less expensive and less flexible computers for less intensive tasks are more than offset by decreased flexibility in deploying these workstations and by increased support costs. With a single desktop platform savings in technical staff time accrue both because a smaller set of computer configurations must be mastered and because technical staff are freed from the need to develop cumbersome, and unstable methods of providing services on less capable hardware.

Armed with a powerful set of development tools and workstations fully capable of supporting graphical user interfaces and network access, development of client applications progressed very rapidly. Two new tools, an object oriented interface builder and object toolkits bundled with our development environment, provided a dramatic jump-start to our effort. These tools fostered an entirely new approach to the application development process. In place of the traditional waterfall model of application development, with its discreet steps of analysis, design, implementation, testing, release, and maintenance, a new iterative approach was followed[4]. In this approach prototypes were created rapidly using the new development tools and users were able to respond to the look, feel, and functionality of an application in the first weeks of a project. Rather than attempting to deliver all conceivable functionality in one pass through the development cycle, a simple version of an application was quickly developed and deployed, usually within three months after the original prototypes. The modular, object oriented composition of application allowed the developers to readily and rapidly extend the functionality of the application in response to user preferences or changing business conditions. This approach of concentrating on core functionality and prototype evolution, along with the added emphasis on abstraction in object oriented design and programming, also greatly reduced the complexity of the problems the developers faced, enhancing their ability to deliver quality products and reducing the risk that the sheer scale and complexity of a problem would jeopardize a project's success. Where two years ago we introduced a new application or a major upgrade to an existing application perhaps twice a year, we now introduce a new application or major upgrade every four-to-six weeks. We estimate that programmer productivity has increased between 3-to-5 times based on comparisons between prior development efforts and analogous efforts in the new object oriented environment.

VII. Staffing Issues

We began our project in 1988 with a small, highly motivated, and skilled team of developers for whom experimentation was both an important tool for discovery and an organizing principle. Addressing a new problem domain with novel technology required a flexible, exploratory approach, and the early of results of our work were encouraging. In a short time we created a functioning integrated data model and database; a new suite of previously unavailable management reports; and a nascent set of network services such as electronic mail and file servers. These early successes reinforced our tacit belief that a small, highly motivated staff could achieve great results. But this structure began to show signs of strain as the scope of the project and the community it served grew. In each phase of our project, making the transition from a small, leading edge development effort to a high-volume production service provider forced us to re-examine our organizational structure.

As our services became more widely used and an infrastructure of networks and servers was building throughout the Division, the need for more formal procedures and methods for planning, analysis, design, and implementation came to the fore. In addition, it became clear that not only did we need a larger staff, but we also needed to depart from our traditional, jack-of-all-trades approach and develop staff specialists. Indeed, this shift toward specialization was our first real step toward a more coherent and mature staff structure. One model for assessing the maturity, or capabilities, of an organization is the Software Engineering Institute's (SEI) Capability Maturity Model (CMM).[5] This model describes the attributes of organizations and their processes at five distinct levels of maturity or capability, beginning with the initial (or chaotic) level (the most

---

[4] The primary method of project management and software engineering that we adopted was the Booch method. A clear benefit to object oriented design and programming, in Booch's view, is the ability of designers and programmers to work with higher level, encapsulated abstactions that are naturalrepresentations of objects in the problem space and which reduce a program's complexity. See Booch, Grady. *Object Oriented Design with Applications*. Benjamin/Cummings, Redwood City, California, 1991.

[5] Paulk, M., Curtis, B., and Chrissis, M. "Capability Maturity Model, Version 1.1." *IEEE Software*, July 1993, pp 18-27.

common), through repeatable, defined, managed, and optimizing levels. In the SEI model,

> Success in [initial level] organizations depends on the competence and heroics of the people in an organization and cannot be repeated unless the same competent individuals are assigned to the same project. Thus, at [the initial level], capability is a characteristic of individuals, not organizations.[6]

Our early staff organization and processes clearly exhibited these characteristics. Moreover, as our project was growing on several fronts simultaneously, we could not simply reassign trusted staff members to new projects, we needed more hands. In response, we increased our staff size and, more importantly, began to segment our staff into specialties. This specialization allowed us to develop deeper expertise in specific technical domains; to improve our institutional memory by assigning whole areas of expertise to individuals rather than spreading the load among several staff; and to more rapidly deliver service by reducing the task switching overhead created by constantly changing the technical domain in which a staff member worked.

Despite this improvement, our organization remained at the initial level of the SEI model for some time. Progress to the next level is characterized by establishing repeatable and predictable processes and, even after instituting specialization, it still remained difficult for us to accurately predict the time or resources required to complete a new project in the absence of formal processes. Instead, individual projects continued to be managed in an ad hoc fashion and, although we were able to provide more services more quickly through specialization, we were no more capable of making informed resource allocations than in the past. After the initial glow of the improved throughput that we gained from specialization began to wear off, the continued difficulties of reliably and consistently repeating successes (or avoiding failures) resurfaced. At this time, we began a short-lived and unproductive search for an "out-of-the-box" methodological panacea. But this effort failed primarily because we had not prepared our organization for these new methods. That is, the fundamental characteristics of our organization remained ad hoc and new methods were quickly abandoned when subject to stress or shifting priorities.

Our second major step toward a more mature and capable organization came after our unsuccessful attempts to find and implement a "magic" methodology. At this time we began to realize that the process of elevating and bringing consistency to the performance of an organization was one of continuous improvement, where fundamentals must be learned and routinized before more advanced and specific methods can take root. Along these lines, we began to develop basic procedural guidelines for application development and system administration processes. The aim of these guidelines was not so much to rigorously define specific procedures, but rather to establish a basic set of operating principles, a framework which could at once guide our current processes and evolve into a set of more formal procedures. In short, this approach reflects the view that radical change is more difficult to achieve in an organization than gradual and steady improvement that first addresses the most basic factors in an organization's capabilities. We believe that our efforts along these lines have raised us to the repeatable level of the SEI model and we are now engaged in fleshing out our framework to achieve the third of the five CMM levels by defining and documenting our development processes. Most importantly, we believe that our organizational capabilities have improved over time, and through the SEI CMM approach we will be able to continually enhance and improve these capabilities.

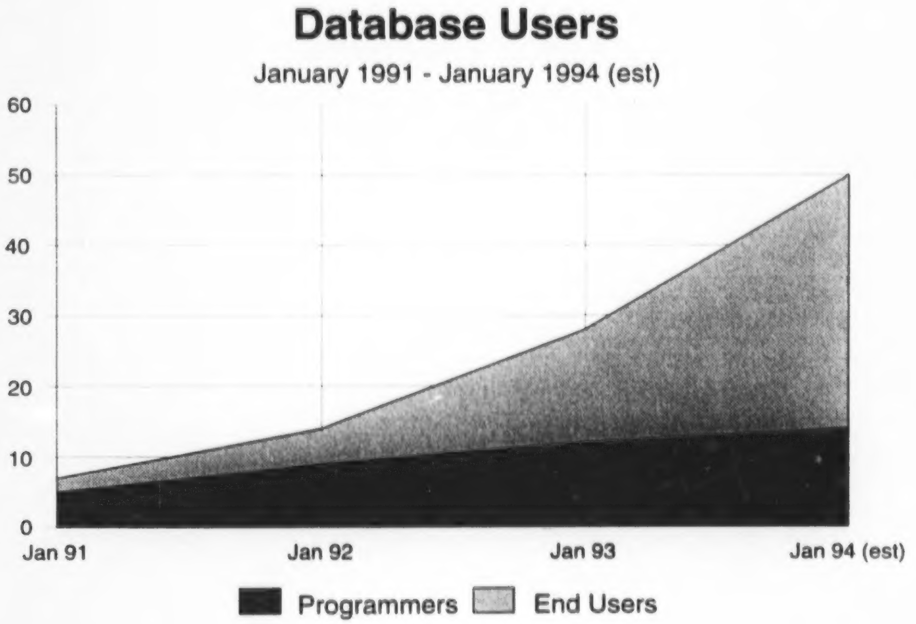
---

[6] *Ibid.*

VIII. Results

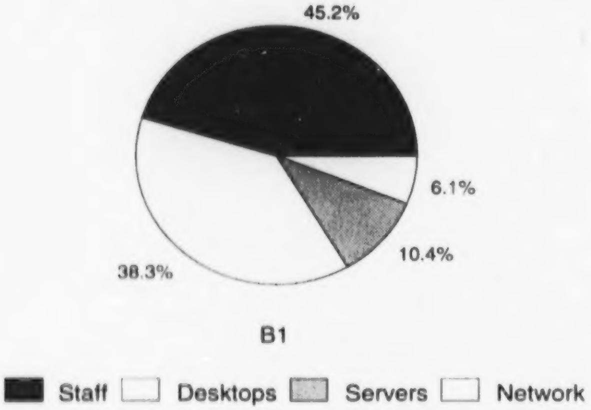
Each of the three major technical components of our architecture are now in place. The core of our integrated database was completed in the second year of the project and has since been enhanced by the addition of new data domains and security mechanisms. Currently, the database incorporates information on academic and non-academic personnel, financial accounting, pre- and post-award grants, facilities management, professional (clinical) fees, course and instructional effort, regulatory compliance, and equipment inventories into a single data model. It is used as the basis for most application development in the Division. The success of the database component of our project has been confirmed by the recent decision of the University to work together with the Division to scale the database to a University-wide scope, providing the University with its first integrated, universally available decision support database.

The infrastructure of networks and file servers grew steadily throughout the project and achieved 95% coverage of administrative users in the winter of 1992. Approximately 250 administrative and 750 academic users in the medical center and nine other campus buildings are now served by the network.



Delivering Division-wide access to our database was one of the major motivations behind the initial expansion of our project, and today over 50 administrators access the database through client/server applications, with 25-30 users typically connected at any one time. Custom applications have been developed to support personnel management (academic and non-academic), facilities management, post-award grant and account fiscal management, regulatory compliance, and unrestricted budget modelling. The role of desktop computers in the success of our project extends beyond custom application development. Service calls among the community of users with the new client workstations has fallen to one half their prior levels, belying the stark contrast between the underlying complexity and power of the new workstations and the character based PCs they replaced. Other tools, such as ad hoc query tools and links between spreadsheet and the database provide users with alternative methods of accessing and manipulating data.

Distribution of Client/Server Costs



The critical role of the desktop computer in client/server technology is reflected in the graph above. Desktop computers accounted for over one third of the ultimate cost of our client/server implementation, exceeded only by technical staff costs, and constituted over two thirds of the capital budget. While desktop computers account for a large the proportion of the total cost, it is important to realize that these costs would generally be incurred anyway. But rather than treating these costs simply as incremental, ad hoc operating expenses, we view them as a major component of an institutional capital plan. We believe that the scale of the expenses associated with desktop computers points up a need for a fundamental change in how they viewed, away from the uncoordinated approach of the past and toward a managed capital investment.

Over the past five years we have attained most of the goals of our client/server development project. By first building a solid infrastructure and then concentrating on client-side development using object oriented methodologies, we are now capable of reliably delivering data and applications to our customers even as desktop computer technology rapidly evolves. Our user base has grown from a handful of senior managers to users at all levels of administration across the Division. Our staff and organization continues to enhance and improve its capabilities and services by exploiting the underlying technical architecture in new ways while continuously striving to improve the reliability and quality of our procedures and methods. Most importantly, however, by incorporating desktop computers into the institutional and capital planning of our Division, we have been able to provide new services more quickly and reliably to our user community by developing a comprehensive technical architecture extending from our server technologies to our users' desktops.

Right Sizing a Mainframe Administrative System Using Client/Server

Presented at CAUSE93 by

Ron Dawe, Assistant Director, Information Systems, Colorado State University

Robert Cermak, Director, College of Business Computer Center, Colorado State University

RIGHT SIZING A MAINFRAME ADMINISTRATIVE SYSTEM USING CLIENT/SERVER

During the last five years there have been extensive changes in the administrative computing environment at Colorado State University (CSU). Beginning with a planning process that developed an open blueprint for the future, the University has moved carefully forward towards an open, collaborative administrative environment predicated on a gradual transition to client/server systems, in house development using CASE, and broad campus involvement in the development of next generation administrative systems.

THE MAINFRAME CONVERSION

In 1991 CSU completed the conversion of its administrative software to IBM hardware and the CA-IDMS database. The current mainframe configuration is an IBM 3090 model 200E with 60 gigabytes of EMC DASD attached. All of the major administrative systems are purchased and, with one exception, are still on vendor maintenance. An IBM RS/6000 model 370 is also being used to support an administrative data warehouse. It is running Oracle version 6 and a migration to version 7 has begun. These systems support the University's administrative needs with good interactive response time and an adequate batch window.

This phased conversion process has not been without its bad days and some trauma, but the campus is generally very pleased with the results. Without this stable environment as a starting point, the transition to the next generation of administrative computing would not be possible.

THE ADMINISTRATIVE COMPUTING PLAN

CSU's Office of Information Systems (IS) initiated a campus-wide administrative computing planning effort in 1990. The resulting planning document encouraged the use of partnerships for implementation and recommended special funding to support such efforts.

"The Plan" was a key document for this project. It provided the high level management support for important changes in the way applications software is conceived, designed, and implemented. Its broadly based focus and extensive emphasis on the larger campus needs were catalysts for a shift in emphasis that continues to benefit the University in more ways than this project highlights.

THE INFORMATION WAREHOUSE EVOLVES

Significantly, the planning document was sub-titled, "Creating a Balanced Environment." It contained a clear call for better vertical balance between both mainframe and desktop platforms and central and college level applications. A key concern for the planners in preparing for this adjustment was the need to create standards for data exchange. This was particularly true given the University's heterogeneous administrative environment of DOS, Macintosh, and UNIX computers.

The initial effort to standardize distributed data focused on developing mainframe data extracts for use on microcomputers. At the time, human resource system extracts were available, but their content had been defined centrally and was in need of review. No standard extracts existed for financial or student data. An open process was launched that brought central data managers and college administrators together to define the content of a complete set of data extracts. This required considerable "education" because most of the new departmental computer people were unfamiliar with the amount and complexity of the data in the mainframe systems. (The classic case of users not knowing what it is but knowing they want it.) The teams were assisted by technical people to assure that the data was suitably organized for use in a relational database.

In parallel with a growing interest in access to central data, departmental microcomputer users were maturing in their use of desktop software. Spreadsheets were being replaced by full featured database products as platforms for local applications, and "software standards" began to emerge. Standards that were not centrally imposed but grew from popular support for a product throughout the campus community. For example, when a critical mass of interest developed for Paradox and Oracle, IS responded to this development by providing support and encouragement. But, this approach created problems because Information Systems did not always have appropriately trained people.. Turning to the source of the interest in these products, IS begged, borrowed and sometimes shanghaied talent from the user community.

Having developed complete data extracts that met the needs of college administrators, IS needed a distribution platform. The first hardware platform for the data warehouse was a 486 Banyan Vines server running Oracle. Departments accessed the data warehouse from their microcomputers using Paradox and the middle-ware products SQL*Link and SQL*Net. As usage grew, the Banyan Oracle server was ported to a more powerful and responsive UNIX minicomputer but not without some discussion of costs. A general rule for expense distribution has existed for many years that stated the departments paid for the desktop system and its connectivity while IS purchased and upgraded the central facilities as needed.

In the early nineties, CSU found itself in a unique situation: a stable mainframe environment, a plan that mandated change, and the appropriate resources available to make change happen.

DEVELOPING A COLLABORATIVE PARTNERSHIP

Coincident with the emergence of the data warehouse, the College of Business Computer Center (CBCC) was asked to develop a management information system for the College. Specifically, the Dean wanted to have pertinent and current management information at his fingertips. Existing University reporting systems could not support this for the following reasons:

> **Transaction processing was not timely:** Most data was provided through manual processing of administrative documents. This processing involved extensive authorization loops and hand carried mail which typically delayed processing one or two weeks.
>
> **Administrative units could not easily track transactions:** Administrative units were removed from transaction processing resulting in a further one or two week period when the exact status of a transaction could only be determined by contacting the University's accounting personnel.
>
> **The University's mainframe applications did not support easy to use query or reporting tools:** Ad hoc query tools did not exist, and, even though information was accessible, the access methods were not suited to the casual user.

It was clear that gathering a timely and comprehensive data set would be the key element to meeting the Dean's request. Post processing data capture from the mainframe systems was discarded as a standalone solution because the authorization loops caused significant delays in mainframe data entry.

Preprocessing data capture of all College transactions was the most feasible solution but had two weaknesses. First, externally generated transactions would not be captured leaving a data set that was incomplete. Secondly, CBCC staff had already experimented with several database products. PC based databases were too weak for the job while minicomputer databases were beyond the limited financial means of the College. Only the relatively new client/server software offered a cost effective solution that still provided enough performance, security, portability, and connectivity to make such an approach viable.

To address the second weakness, the CBCC opted for an Oracle client/server platform built around the College's existing Banyan Vines network. Key factors in this choice were Oracle's portability, availability for Banyan Vines networks, and the Oracle expertise already present in two other administrative areas on campus. With hindsight, this was an excellent choice as the product has met all expectations.

Addressing the first concern of preprocessing data capture was a more complex issue, but, with the Dean's support, the CBCC proposed a campus wide effort to develop electronic transaction initiation, approval, and processing. This project was presented as the College Information System (CIS). A set of five integrated client/server database modules (Human Resources, Financial Services, Undergraduate Information, Graduate Information, and Research Services) that addressed the general needs of administrative end users.

The CBCC had three objectives in proposing the CIS:

1. Make all campus transactions accessible to all users.
2. Promote changes in administrative procedures that led to more efficient processes.
3. Create a broadly based coalition with sufficient political muscle to manufacture change.

This strategy was initiated by contacting the College Administrative Advisory Group (CAAG). The CAAG is a group of Assistant Deans, Directors, and Managers from the colleges, libraries, continuing education, and several auxiliary agencies. This group agreed to endorse the CIS if the University's Information Systems was an active participant in the design and implementation of the project. After joint discussions, the CAAG, CBCC, and IS signed a partnership agreement with the following distribution of responsibilities.

Through its broad support and direct channel to the Deans, the CAAG empowered the partnership to approach and enlist key university administrators from Internal Auditing, the Controller's Office, Registrar's Office, Human Resource Services, and many other areas. Additionally, the CAAG provided end user expertise, prototype reviewers, and assumed the financial responsibility for connecting their users to the CIS.

Information Systems enabled the partnership by providing financial and technical assistance to the CBCC staff, reworking rough CIS prototypes into production modules, providing the production platform and server software, and guaranteeing to maintain and enhance the CIS production modules. Their successful efforts to tactfully defuse the inevitable political fallout were also critical to the project's overall success.

CBCC implemented the partnership by providing client/server expertise IS had not yet developed, a development platform, and strong, aggressive team leadership for module analysis, design, and prototyping.

METHODOLOGY

A rapid prototyping methodology was selected for the following reasons:

> **Rapid prototyping is an action oriented process that discourages posturing and gamesmanship.** Prototyping brought tangible products into the hands of reviewers quickly. The speed of delivery made review groups intolerant of group members who dissembled or otherwise blocked action items.

Prototyping let the end user shape the design process. Since the design process was not artificially structured by the development team, the end users had a greater sense of control, involvement, and impact.

End users were not familiar with the potential of the client/server environment and had difficulty visualizing what was feasible. Prototyping's iterative review process allowed end users to experience client/server possibilities and encouraged a broader, more visionary response from end users.

Prototyping built a working model that shaped reasonable end user expectations. End users reviewed a working prototype containing the features they had specified. If a feature could not be implemented, the reasons were identified and explained to the end users during the review process.

ANALYSIS/DESIGN OF BUSINESS PROCESSES

Analysis and design committees were formed. They consisted of three groups: **policy setters** who managed many end users, were responsible for a family of business processes, or influenced the University policies and procedures surrounding a family of business processes; **end users** with generally recognized expertise within a specific business process; and **technical experts** who translated and reconciled committee work into prototype specifics. Policy setters were typically University business officers, for example the Controller, Associate Dean of the Graduate School, or an Assistant to the Dean for Administration (College). End users came from all parts of the University organization, for example accounting technicians, undergraduate advisors, or staff assistants. Technical experts were managers of administrative subsystems, development team members, or analysts from Information Systems,

Analysis and design success was predicated on an environment that encouraged the free flow of ideas, mutual respect for each group's needs, and dialogue instead of confrontation or impasse. Needless to say, the committees struggled to maintain this ideal, but leaders from each group stepped forward at crucial times to facilitate, encourage, and defuse. This ongoing and relentlessly tiring task proved to be the most difficult of the entire project.

As the committees defined business functions, the technical experts translated these definitions into database design, functionality, and finally working prototype. Committee results were relayed to the programmers immediately after the committee meetings. Any concerns or questions about technical feasibility were identified and brought to the committee's attention at the next meeting.

The programming staff consisted of one or two graduate students who were usually quite familiar with the Oracle development tools. They were equipped with powerful PC workstations and the standard Oracle development tools. They worked an average of

20 hours a week with an increase to 25 hours a week during the end user review process.

The development team set two design constraints for the committees.

First, the prototype would contain 80% of any business process and would not address the remaining 20%. This constraint was based upon the belief that the easiest 80% of any business process contained all the essential elements while the remaining 20% contained all the customization (and 80% of the development workload).

The CAAG and Information Systems did soften the long term impacts of this constraint by adding two stipulations: 1) Any administrative unit could do its own customization but had to bear the full cost of that effort; 2) A standing review committee would judge future customization proposals and those that provided broad benefits without undue costs would be adopted and maintained by IS as part of the production module.

The second constraint stated that manual processes would be mirrored whenever practical. This constraint had four powerful advantages: it reinforced the value of existing processes and their supporters; it reassured employees who were dependent upon the existence of these processes and would be important contributors to the prototypes; it encouraged the committees to focus on how they might improve existing processes without reinventing them; and it helped overcome end user inertia by producing prototypes that contained many recognizable elements.

Even with the second constraint, the open environment of the analysis and design committees encouraged an atmosphere that has lead to truly remarkable changes in administrative perceptions. These frank and open discussions of administrative procedures have led to a re engineering of administrative processes that goes far beyond the scope of the initial project. In fact, all of the campus' administrative procedures are being looked at from a new perspective. A perspective that focuses on using collaboration, cooperation, and partnership to effectively achieve improvements in quality and productivity. Improvements that are being defined by the whole campus community.

IMPLEMENTING THE PROTOTYPE

When the end user analysis/design committees were finished and a prototype was working, each administrative unit was asked to provide a review group consisting of 4 to 6 end users. Most of these end users had not actively participated in the analysis/design process but were going to be primary users of the production module.

Module prototypes went through six to twelve review sessions depending upon the number of available reviewers and the complexity of the module. Reviewers came to a special facility where they had their own PC for testing the prototype. To help the reviewers focus on the prototype features and to reduce frustration with the new environment, there was one technical support person for every two reviewers.

As each feature of the prototype was explained, the reviewers used the feature and provided their perception of its effectiveness. Every effort was made to address ideas, suggestions, and problems on the spot. Most of the comments were refinements to screen or report design, but there were instances where reviewers made suggestions that offered significant improvements to the functionality of the prototype. These suggestions were shared with other reviewers. If they reacted favorably to the idea, it was presented, informally, to the members of the end user analysis/design committee. Several productive enhancements were identified and implemented in this manner.

All review groups were given several weeks to comment fully about the prototype. During this period, all modifications were implemented and the prototype was documented. In the final step, the prototype was handed over to Information Systems to be reworked and reshaped into a production module.

TRANSITION TO THE UNIVERSITY LEVEL

The prototypes created by the College of Business were transferred to IS for implementation across all colleges. Initially we expected that the modules would be implemented as created in the College of Business. However, when the time came to do it, we found the initial module prototype contained unacceptable technical differences regarding data and keyboard standards. These differences were resolved in future prototypes. Nevertheless, there were real difficulties in implementing the prototypes from an expected source but a surprising direction.

The first module (graduate information) was shared with the graduate school and there were some significant differences between the perceptions of college administrators and the graduate school. Superficially, these differences focused on some of the traditional reporting, but at a deeper level there were real differences in defining the roles of the respective players. This was the first time a central group had not dominated the definition and design process for central administrative software. Overcoming these differences required actively selling the attractive functionality of the client/server modules.

As the second module (undergraduate information) was being completed and receiving rave reviews by college administrators, central student records administrators took notice. It was obvious that the module contained degree audit functionality that was planned for the mainframe. After a detailed review of the module, it was agreed that the college created software would be enhanced to produce an official university document. This also meant that central policies would be reflected in the module and that the records office would maintain most of the data tables. The addition of central influence meant that the project had changed from the original plan.

REDEFINING ROLES

The desire of central administrators to use parts of the CIS system was largely unanticipated and led to the creation of a supplement to the informal agreement created at the beginning of the project. The supplement dealt with the process of moving the modules from prototype to production and specified the role of the college administrators. It also clarified respective roles regarding future enhancements, maintenance, and any new modules that may be created by the colleges. A college review team was established to represent the interests of the colleges in the implementation process. Check points were defined to assure the integrity of the functionality in the prototypes and a thorough college level testing process was included. A key part of this document was the role of the college review team in reviewing and prioritizing requests for change.

However, this supplement to the contract was limited by an agreement with central administrators on three issues:

1. The availability of data must be approved by its central owner.
2. Any process that creates data for input to a central application must be approved by the responsible central administrator.
3. Both the content and presentation format of data for students must be centrally approved.

These limits are not surprising. What was surprising is that these are the only limits. In addition, the agreement specifically states that central administrators will not judge the desirability or usefulness of college developed software.

A simple process was created for implementing the supplement. Namely, Information Systems will assure that the appropriate people are brought together to review any proposals from the colleges. So, after several years of organizational jostling, a compromise acceptable to all parties is in place. The colleges have established their right to define and create centrally supported administrative software. Central administrators have established that they control certain aspects of university administrative data and its use. Progress.

IMPACTS ON INFORMATION SYSTEMS

These interlopers in the colleges had their impact on the central development staff, too. UNIX - which one? Relational database -what's wrong with IDMS? Client/server - sure. Come back after we get payroll fixed. Seriously, this project was the opportunity to make some significant changes within the central Information Systems office. The Information System's staff was very much aware of the changes occurring in computing, and the only real question was what would the specific impacts be.

Whenever IS encounters significant change, it is always the policy that existing people will be trained to take on the new tasks. The only question is how to do it. For this project, the start was to transfer a software developer from an administrative department to Information Systems. This person had extensive experience with microcomputer applications development. Consequently, he installed the first module and became the seed person for training the mainframe staff.

Unfortunately, until very recently all of the central support skills for the project were focused on only one, sometimes two, people. The new employee was the systems programmer, DBA, network manager, and the applications specialist. His vacations were scary events. IS' efforts to train other analysts to support the project did not yield any clones because it was simply too much for one person to learn all at once. The new analyst's experience represented ten years of incremental learning, and the mainframe analysts were simply overwhelmed. Early attempts to create micro-generalist clones were also handicapped by a lack of formal training. Consequently, a different approach was necessary.

Will any one be surprised that the new functions are now being dispersed to their mainframe equivalents? And formal training classes are being provided where the expense is not great. The IDMS DBA is going to Oracle school. A mainframe systems programmer is going to UNIX school. These two will use a recently replaced IBM RS/6000 as a training platform. Security administration on the CIS platform will be handled by the same people who manage mainframe security. Applications support will be provided by the same analysts who support the mainframe. Unfortunately, providing a complete Oracle training program for all of the analysts is not financially possible, so their training will be on-the-job implementing specific projects using training materials supplied by the vendor: tutorials, manuals, and texts.

One problem remains. Within Information Systems, the skills required for designing and creating new applications have fallen into disuse because of the policy of purchasing mainframe software. In addition, the contemporary methodologies are quite different from those used ten years ago. CIS implementation has become an opportunity to explore these new technologies. Accordingly, an experiment with Oracle CASE is underway. The prototypes created by the College of Business are viewed as the strategy and analysis phases of information engineering. The detail design and construction phases will be done using Oracle CASE. Support and future enhancements will use the CASE dictionary and Oracle tools rather than relying on traditional documentation and third generation languages. The early results of this CASE experiment have been very promising.

COMPUTER DEMOCRACY

Technologies often go through phases that start with only a few highly skilled people users and end with the product becoming ubiquitous. The telephone is a commonly cited example, and a parallel could be drawn between the early telephone operators and the mainframe programmer. The CIS project at CSU may be the technological turning point where administrative computing is moving from an autocracy towards computer democracy. Obviously, the project has distributed many of the key aspects of administrative computing to a far larger group of people. However, the most important accomplishment may be the general acceptance of distributed responsibilities that has followed the distribution of tasks. The acceptance of these new responsibilities will be the foundation for further democratization of administrative computing at CSU.

Security in a Client-Server Environment

Gerald Bernbom
Mark Bruhn
Dennis Cromwell

University Computing Services
Indiana University

Paper presented at CAUSE93
December 9, 1993
San Diego, California

> *"Securing a client/server environment is like throwing mud at the invisible man -- it may be messy, but pretty soon you get an outline of what you're up against."*
>
> (James Daly, *ComputerWorld Client/Server Journal*, August 11, 1993)

Security: the basic message

There are three basic messages that we want to communicate in this paper. First, that the purpose of security is to enable access, not to impede access. Our approach to the design of security solutions is focused on delivering access to computing and information resources at levels of risk that are known and accepted. The more access we want to deliver, the more attention we pay to security. Client-server computing opens new paths of access, and these require new security solutions.

Second, that there is no single solution to information security. The metaphor of a locked door -- that if we can just put the right lock on the door we'll be protected against intruders -- no longer applies. A complex computing environment, especially one that includes a client-server computing component, presents multiple points of entry: the workstation, the network, and local and central servers. The challenge is to recognize these points of entry, to understand and assess the risk that each presents, and to choose protections that are prudent and cost-effective.

Third, that even if specific security solutions are tactical responses to risk assessment (and they most frequently are), the process of designing and implementing security solutions is based on principles, objectives, and an overall strategy.

Information Systems at Indiana University: a two-minute tour

Security solutions are always in response to specific problems of access and risk. A brief overview of Indiana University's computing environment for enterprisewide information systems will help form the basis for understanding the design of our security responses.

The network is the integrating force in our computing environment; it ties together an array of computing and information resources, and is the bridge between central computing facilities and individual workstations or departmental networks. IU operates a multi-protocol network, carrying the TCP/IP, IPX, DECnet, and Appletalk protocols. User workstations at Indiana University are a mix of Intel-based DOS and Windows personal computers, Macintosh computers, and a small but growing number of Unix workstations.

The primary host computer for enterprise information systems has been a large MVS mainframe. To this we have recently added Hewlett-Packard application hosts running Unix (HP/UX) for client-server systems. We also run a large VMS cluster, primarily for instructional and research computing and as a host for IU Bloomington's campus-wide information system (though our CWIS is rapidly migrating to client-server technology with Gopher and World-Wide-Web).

The database management systems we use for enterprise information systems are DB2 on the MVS host, and Sybase on Unix application hosts. We also run Ingres, again primarily for instructional and research computing.

Our application development and CASE tools are Uniface and Bachman. Uniface is an application development tool for creating client-server and host based applications. Bachman is used for data modeling, process modeling, and database design.

The computing environment that is the target for our first major administrative information system using client-server technology consists of: a Hewlett-Packard (HP/UX) host, the Sybase database management system, applications written in the Uniface development environment, TCP/IP connectivity, and client software running on Windows or Macintosh workstations.

The application that is the target for our first major client-server system, and thus the first iteration of a client-server security design, is a university-wide financial information system. We have explored issues of client-server security with two smaller systems that we have used to pilot new technology, but the security risks of a distributed financial information system -- entry of financial update transactions at their source, routing and approval of transactions by multiple users, and eventual posting of transactions to the general ledger -- required a more comprehensive security response.

Security: principles, objectives, and strategy

Security principles apply to all stakeholders in an information systems implementation effort: application developers, users, security managers, and university administrators. There is virtually no such thing as the elimination of risk in a computing environment. If there is a resource, and there is access to that resource, then there is risk to the resource. The principles of security that we apply are risk analysis and risk reduction, with the intent to manage risk at an acceptable level. What everyone involved in an information system implementation must understand is that the final design will entail risk. The responsibility of these stakeholders is to understand the risks, understand the steps taken to reduce risk, and accept the results.

Security objectives provide users, developers and security managers with a way to focus their analysis and attention on broad, general areas of risk. The primary objectives of a security analysis and response are:

- User identification. Knowing who the user is; assuring that each user can be uniquely identified.
- User authentication. Knowing that the user is who s/he says s/he is.
- User authorization. Knowing what is permitted or prohibited to each user, and enforcing these permissions and prohibitions.
- User accountability. Knowing and keeping record, for each access or other significant event in a system, the identity of the user responsible for that event.

A *security strategy* provides the framework for the development of an overall security design. It is the strategy that brings continuity and helps assure forward progress to the security efforts of an organization. The information systems security strategy that we use at IU has four key components.

Security design is *iterative*. We engage in a cycle of identifying security exposures, assessing the relative risk of each exposure, designing an intervention to reduce the highest risk exposures, evaluating the effect of the intervention, and identifying the remaining exposures.

Security design is *collaborative*. There are multiple stakeholders, both within the computing organization and among the user population, who have an interest in the design of security solutions. Because understanding and acceptance of risk is the basis for a security solution, these stakeholders must participate in design process. There are also multiple areas of expertise needed to identify exposures, assess risk, and design interventions. In increasingly complex computing environments, the need for collaboration across several areas of technology specialization becomes greater: network designers, network operations staff, workstation software specialists, database administrators, in addition to application developers, user support staff, and security management.

Security design is *responsive*. Fundamental technology components are changing on an annual basis, if not more frequently. Changes in technology may open new exposures that did not previously exist. Or new technology may create opportunities to respond to exposures that had been previously left unaddressed. Security management and its collaborators from other technology areas need to monitor change in the industry, to assess the effect on risk and on the available measures of protection.

Security design is *accumulative and evolutionary*. Each security action is a response to a specific exposure or set of exposures; it is an intervention designed to reduce some known risk. As such, security actions are components of an overall security solution, but no total solution is ever implemented. One of the greatest challenges in security design is to choose components that work together, and that minimize the constraints placed on future choices of security actions. Equally challenging is to choose security components that fit with, or anticipate, the direction of the industry on providing information systems security solutions.

Securing the mainframe in an open environment

These principles and strategies were initially developed and refined in the design of security solutions for our mainframe computing environment, especially as we expanded access to this traditionally closed environment to a wider audience of university users. A brief overview of the migration of our mainframe connectivity from a relatively closed SNA network to a more open TCP/IP network will set the stage for the more radical transformation we are responding to in the area of client-server security.

At most institutions, security of the mainframe environment is relatively mature, and there is abundant experience in implementation and administration of the various host access control products available for these computers. The Indiana University situation in this area is typical: mainframe security has been the focus of our attention for some time. CA-Top Secret and Terminal Productivity Executive (TPX) have been installed for several years, and are interfaced to provide user login and menu services, password authentication and management, scripted application logins, and access authorization. CA-Top Secret is also integrated with many other program products to limit multiple application user data bases, which helps reduce administrative overhead.

Outside of these standard host access and authorization requirements, we also had some specific objectives to consider in providing access to the mainframe over a more open network:

- We had to offer "guest" access to an otherwise secure computer (e.g., to provide anonymous and unlimited access to the library's on-line catalog).
- We had to cope with unpredictable connections from diverse users on the same "open" network.
- We wanted to protect passwords on the network as much as possible.
- We needed to improve on password as the sole user authentication method.

In order to satisfy these objectives, computing staff members from data administration, security administration, and network operations spent many meeting hours analyzing the network topology for possible exposures. In the end, we addressed these concerns with four modifications:

- We established two "access areas" on the mainframe, each with its own network interface. One area permits access to only "guest" services, such as the library on-line catalog and some student-oriented applications; the other area permits access to all defined applications. We used CA-Top Secret and TPX to enforce identification (login) and password management policies, and to limit the applications that could be accessed in the "guest" region of the computer.
- In conjunction with these separate access areas, we installed router filters that permit access to the secure access area only from a select set of networks within the IU domain, and that deny access from specific high-risk networks (e.g., campus public computing facilities).
- We implemented password token cards as an additional method of user authentication for users accessing the secure area. Each card is keyed to an individual user. It is used in a challenge/response dialogue during the system login sequence and must be in the possession of the user at that time. The combination of these two authentication methods -- something the user knows (password) and something the user possesses (password token card) -- is generally accepted in the industry as adequate for all but the most sensitive systems.

(Figure 1 gives an overview of this mainframe security configuration.)

We feel comfortable that our efforts in these areas have resulted in adequate protections for the mainframe environment. However, we still must contend with the constantly changing software set and various network topologies in order to ensure that changes to the mainframe and network environment do not adversely affect security mechanisms.

Client-server security

The client-server environment is new to almost everyone. It is a new way to provide access to the same data, stored in a new place, in a possibly new format. But there are still the same security requirements that were encountered in the mainframe environment. Security administrators, application developers, and system managers must still have the same comfort level in user identification, authentication, authorization, and accountability.

As designers of a client-server security architecture, starting basically from zero, we agreed on some basic understandings:

- All of our solutions may very well be interim ones.
- We must always plan for enhancements or replacement based on new software, changes in application requirements, changes in server configuration, etc.
- The mechanisms that we deploy should be able to protect against what we perceived as the highest risk exposures, both in terms of the degree of damage that *might* be done and the probability that the damage would actually occur.

Given these basic thoughts, we developed five core objectives for our client-server security design:

- protect host passwords;
- reduce exposure to network intruders;

- require the same challenge/response password tokens used for mainframe access;
- protect database server passwords; and
- restrict database server access to authorized connections.

There are several components that comprise the security architecture we have developed to satisfy these objectives: client application, network filtering, host security, Security Server, Gateway Server, and Telnet Server.

The client application performs two main security functions:

- it interacts with the host security process and provides the user interface to the challenge/response authentication dialogue; and
- it encrypts the user's host password during identification and authentication so that it never passes on the network in clear text.

A standard client module has been developed to execute these functions. This module can be called from any Uniface client application running on a desktop computer.

The network filtering element of the architecture is comprised of subnet router filtering and a subnet bridge. The router filtering is modeled on our use of network control of access to the secure mainframe region; the router filtering denies all connections to the host server from non-IU addresses and from high-risk addresses within the IU domain. The subnet bridge is placed directly in front of the database server on the host computer; it denies ANY network connections to the host's database server port.

The host security component involved the conversion of our HP/UX operating system to a "trusted system". This irreversible conversion (provided with the operating system by HP) involves the use of a shadow password file and the installation of an audit server, which permits full auditing of users or events.

The Security Server is the heart of the security architecture. This program interacts with all other components, and is the "authorizing agent" for access to the database server. For client application sessions, this host-based server receives, decrypts, and validates the username and password from the client application. If the supplied password does not match, a negative return code is passed to the client application. For both client and telnet application sessions, the server obtains a unique session challenge from the authentication software, and passes it back to the application for presentation to the user. The application then returns the user-supplied response, which the Security Server validates with the authentication software. Given that the challenge/response validation is successful, the Security Server generates a one-time database server password, accesses the database server and changes the user's password to the new one-time password, and writes the new one-time password and a session ticket to a database.

The Gateway Server manages access to the database server. All access to the database server MUST come through this program (the database server port is blocked!), and only with the "permission" of the Security Server. This Gateway Server intercepts connection requests to the database server, and searches a ticket database for a valid access ticket issued by the Security Server. Given that a current ticket is found, the Gateway connects the user to the database server with the one-time password that was issued by the Security Server and stored (encrypted) with the access ticket. Subsequent traffic for the user session are passed by the Gateway Server directly to the database server.

Although the applications developed for the client-server environment are primarily meant to be accessed from a client workstation, we also had to provide for a host-based version of the application for users without adequate devices to handle the client code. The Telnet Server uses the standard telnet service of HP/UX, and is invoked when users telnet directly to the host for host-based applications or other database access tools. The telnet service has been bundled with an interface to the Security Server as well as a menu structure. Following standard host login validation, the interaction with the Security Server provides the same challenge/response dialogue that the client user undergoes, and issues a database server access ticket and one-time password for the user session. The menu structure serves to limit user access to the HP/UX system prompt, and adds convenience for the user when choosing applications. The options on this menu vary with the user: some have only user application choices, others have DBA-oriented tools, such as Interactive SQL. In any case, applications on this menu which access the database server must go through the Gateway Server, which will first check for a valid ticket in the ticket database before connecting the user to that database server.

By way of review and comparison of the security architecture components with our stated objectives we see that:

- we have protected passwords on the application host by encrypting passwords at the client and by using the host's shadow password file facility;
- we have reduced network exposure by using network router filtering to limit the source of connections to the application host;
- we require the use of challenge/response password tokens for all accesses to the application host;
- we are protecting database server passwords by issuing one-time passwords -- which are never known by the users -- for database server access; and
- we are restricting database server access to authorized connections by denying direct access to the server port, and by requiring all other access via the Gateway Server.

(Figure 2 gives an overview of this client-server security design.)

It's worth noting that our client-server security design is a evolutionary development of our mainframe security design. The network filtering of traffic to the application host is borrowed directly from our mainframe security design, as is the use of challenge/response password token cards. In fact, our choice of vendor for password token cards was based on the requirement that a user be able to use the same physical card for authenticating his/her identity on multiple host computers.

Security: the state of the industry.

> "No significant headway has been achieved in any of the competing visions of enterprisewide security..."
>
> "It is left...to the user to build together the available technologies with sound business practices to guarantee the integrity of business information."
>
> (Gartner Group; "Client/Server Security"; *Third Annual Symposium on the Future of Information Technology*; October 4-8, 1993; Orlando, FL)

Our experiences in designing security solutions for a client-server computing environment are consistent with this view of the industry that the Gartner Group offers. There is, among the vendors we have worked with, no shared vision of a heterogeneous client-server security solution.

The database and software tool vendors we have reviewed and worked with offer basic security services, with much attention focused on the problems of authorization services: increasing the functionality of roles and groups, for instance, as a means of more easily managing the grant and revoke of database permissions. By contrast, the database and tool vendors have spent less effort on authentication services, which are often incomplete and need to be supplemented with outside help: either third-party or home-grown add-ons. Unfortunately, vendor emphasis is weighted toward proprietary security solutions: looking for answers within the constraints of their product offerings, rather than helping build solutions that cross these lines. Although their products are "open" in many respects, they are slow to adopt emerging security standards and are surprisingly closed when it comes to enabling software integration with products from other vendors or with user-written code. This mix of minimal solutions for user authentication and an unaccommodating attitude toward external software has made development of high-quality authentication services a particularly difficult challenge in this multi-vendor client-server environment.

Our experience is that the hardware and operating system vendors are doing a somewhat better job on security. They seem to have a good awareness of security issues, and are improving their solutions to problems of auditing, accountability, and system integrity. It is the hardware vendors, too, who have put the strongest support behind OSF/DCE, which presents the best potential as a standard for supporting a heterogeneous client-server security environment.

Responding to the industry

There are three ways in which computing organizations can respond to the state of the industry:

- Assemble its own client-server security solution.
- Design with the future in mind.
- Respond directly through collaboration and market pressure.

Since no vendor or group of vendors -- whether of hardware or software, of mainstream or specialty products -- offers a solution to heterogeneous client-server security that can be purchased and used, the only viable answer today is to assemble a security solution from a mix of purchased and locally-developed components.

In our first iteration of a client-server security design this consisted of:

- Selecting *specialty products that fulfill specific needs* in the computing environment and for the target application. (In our case we chose a specialty product, Unix-Safe, to offer challenge/response authentication on a Unix host computer.)

- Using *features available in primary products*. (In our case, we used the "open server" and "open client" features of our database product to develop a Gateway Server that validated user connections against a valid-ticket database.)

- Using *home-developed code to tie the pieces together*. (Our Security Server is a locally-written piece of code that interfaces to our Unix-Safe authentication product, interacts with stored procedures in our database product to set passwords, and writes the valid-ticket entries that our Gateway Server uses to permit database access.)

Given the developing state of the industry for distributed computing, any client-server security solution should be designed with the future in mind, acknowledging that the security design will be undergoing change for some time to come. One area to anticipate change is in the future features (announced, promised, or merely rumored) of existing software products. A second area to anticipate change is the potential adoption of standards-based features, such as those in DCE Security Services, by hardware and software vendors.

The future availability of these features should be considered in the initial security design -- postponing inclusion of the feature altogether or, if the feature must be locally-developed, isolating it in the design so that a commercial product or standards-based design may be more easily substituted in its place. For example, in our first design of client-server security we include a ticket-database which has interactions with the Security Server (the source of tickets) and with the Gateway Server (the user of tickets). If an industry standard for authentication tickets is adopted by any of our vendors, our design would permit us to replace this initial ticket management system with one that is standards-compliant.

A final course of action available to every computing organization is to create market pressure on vendors to adopt security standards and address client-server security needs in their products. They can make their case to vendors, arguing the need for security standards, and they can take their business to vendors who are willing to work with customers on security solutions. Organizations may do this individually or, more effectively, in collaboration with others.

Toward this end, the Big Ten computing directors have collectively endorsed the OSF/DCE standard for distributed computing and are focusing their attention on influencing a key group of hardware and software vendors. One important way to influence vendors is to place security requirements prominently in all RFPs for client-server hardware and software. Compliance with standards or a commitment to work on an integrated security solution should be a heavily-weighted factor in the evaluation of any vendor's product. Indiana University used OSF/DCE compliance as a major criteria in its RFP and evaluation of host/server hardware for the client-server financial information system.

Conclusion

Indiana University is in the very early stages of implementing a security design for client-server computing that can be applied to enterprisewide information systems. The design we have today is a package of individual security actions in response to known exposures. Our view is that all client-server security designs will, for the foreseeable future, be a collection of such tactical security responses, and that our design will change and evolve in significant ways over the coming months and years. Our confidence in this initial security design is based on our strategy of iterative risk reduction and evolutionary growth; because we are addressing exposures in a planned way and are at the same time planning for change, we feel our first step is a step in the right direction.

Note: The work of two University Computing Services staff members needs to be acknowledged in this paper. Charles McClary (Information Technology Analyst) and Tom Davis (Security Analyst) have done significant research, detail design, and code development on our first iteration of a client-server security solution. Their initiative and individual efforts were essential to the overall success of this project.

Mainframe Security

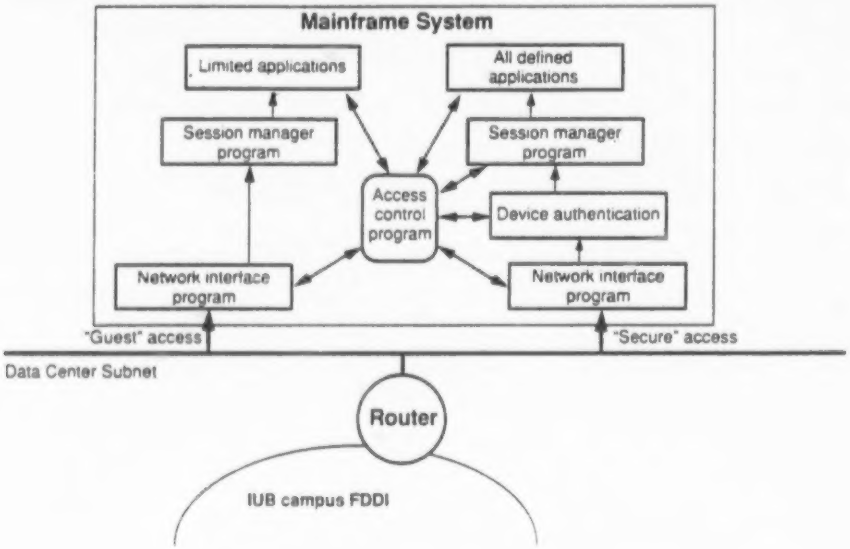

Mainframe System
Limited applications
All defined applications
Session manager program
Session manager program
Access control program
Device authentication
Network interface program
Network interface program
"Guest" access
"Secure" access
Data Center Subnet
Router
IUB campus FDDI

Figure 1: Mainframe Security Design

Client-server Security

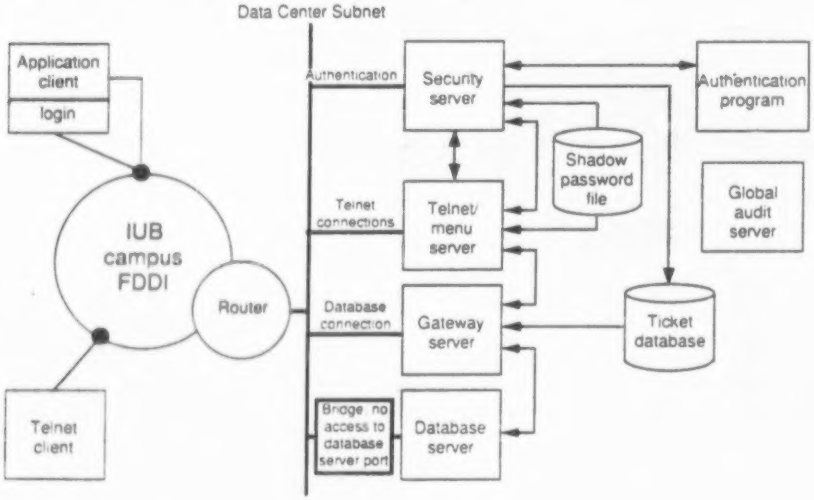

Data Center Subnet
Application client
login
IUB campus FDDI
Router
Telnet client
Authentication
Telnet connections
Database connection
Bridge no access to database server port
Security server
Telnet/ menu server
Gateway server
Database server
Shadow password file
Ticket database
Authentication program
Global audit server

Figure 2: Client-Server Security Design